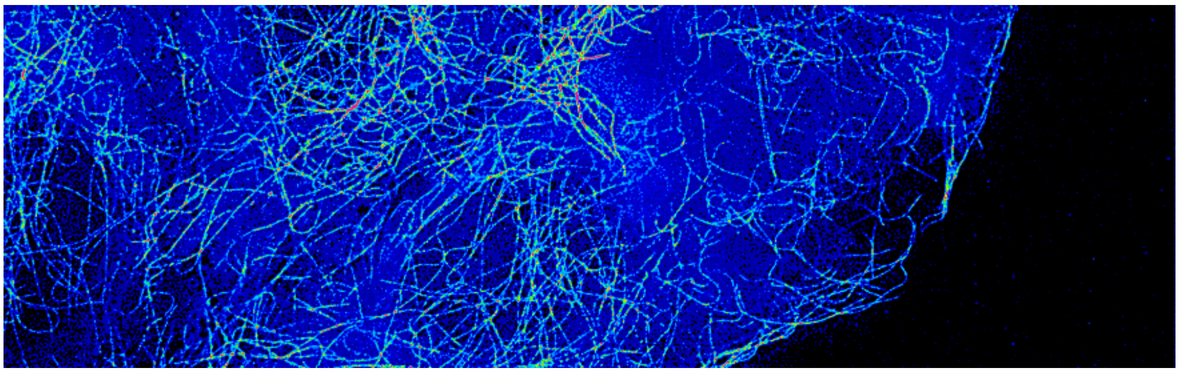


# Enhancing high-density single-molecule super-resolution imaging

**Inês Martins Cunha**

Thesis to obtain the Master of Science Degree in

**Biomedical Engineering**

Supervisors: Prof. Ricardo Henriques
Prof. João Miguel Raposo Sanches

**Examination Committee**

Chairperson: Prof. Maria Margarida Campos da Silveira
Supervisor: Prof. Ricardo Henriques
Member of the Committee: Dr. José Miguel Rino Henriques

**September 2023**

# Preface

The work presented in this thesis was performed at Instituto Gulbenkian de Ciência (Oeiras, Portugal), during the period of October 2022 - August 2023, under the supervision of Prof. Ricardo Henriques.

# Acknowledgments

My first big thank you goes to Ricardo Henriques, my supervisor. Ricardo, I will try my best to summarise everything I have to tell you. I joined your lab as an intern (having no idea what I'd gotten myself into), with zero expectations. I never in a million years thought that I would change as much as I did (for the better) or learn as much as I did. I don't think you are aware of how much you taught me in the last year. Not only about science, but also about research, and life. I will forever cherish and look up to our conversations where you shared a lot of your wisdom/ advice with me: "be the sneaky fish!"; "never stop being unique"; "let's go through the exercise of taking a step back and thinking if this is really the best approach"; the importance of understanding the "why" and telling a story seeing things from the main perspective; 5 second rule for Figures scheme; the catch phrases "I'm not stressed, I'm excited", "Go big or go home"... Ricardo, you gave me the opportunity to participate on events I will never forget. You let me and encouraged me to be an instructor at an EMBO course and at the Abbelight workshop; to go to multiple conferences (SMLMS 2022, NEUBIAS, SMLMS2023); to go to Cambridge to visit labs. This is something not everyone would do for a student, and these experiences truly changed the way I think and how I see the world. Throughout this journey, you gave me so much confidence – I never had anyone who believed so much in me, let alone someone at your career stage. I can only thank you and hope we cross our paths again in the future. I like to believe our relationship goes way beyond a student-supervisor: I see you as a life mentor, and a friend. I will never forget the amazing time I've had in your lab. Muito obrigada por tudo.

Bruno, you already know all of this, but I want to thank you for everything you taught me and for the person you are. We started working together and now I strongly believe I can count on you for much more than work-related stuff. You are an amazing teacher, mentor, friend. I still cannot believe how much patience you have for all my questions ("but how do computers work???", "what is compiling code??".... (I'm sure I've asked you over 100 of these)). You are one of the people I look up to the most, and I sincerely hope our career paths cross again at some point. Now, to António: I could not be happier about you joining our little NanoPyx team this year. I never thought we would have so many things in common and will never forget our conversations (you know the ones I'm talking about). I see you as a great friend and feel very comfortable around you – I feel like I can confess to anything or ask you any topic. You are such a great person, and, as with Bruno, I really hope we can get to work together again. To both of you, Bruno and António, I never thought working could be so fun. As I drove to IGC, I couldn't wait to join you for a coffee and then for some "coding together" sessions. Developing NanoPyx with the two of you was truly one of the most intellectually exciting and fun periods of my life. You made it so much fun – both the working and all our little breaks – and I could not thank you enough for it.

Hannah, you were the first person who introduced me to SMLM. I will always cherish all the patience and availability you had in answering all of my (sometimes ridiculous) questions (examples: "wait... is a ratio of 1:10 equivalent to 10%:90% in concentrations???"; "but how are lasers aligned???"; "what actually happens when electrons go into a triplet state... ?"...). Thank you for everything you taught me.

Esti, I want to thank you for all the life lessons you gave me. We've had conversations you probably don't remember anymore but I constantly think back on – they made me question and form my own perspectives. Besides these, you also taught me so many scientific lessons: "you need to mathematically prove your data can be fitted by that model!"; I remember asking you what is a backend and frontend or what are Fiji macros and how to write them. The insights and opinions you give on scientific research is something I won't forget, and also helped me shape my own opinions. I will always look up to your way of working and critical thinking.

Mario, putting into words how much I have to thank you is extremely difficult. You were the person that influenced me the most, and for the better. We never worked together but we supported each other whenever we needed. Having you around at work immediately made all my days better. Whenever you came to margem sul to visit me, I had some of the best times of my life. You are an outstanding human being, and I will be forever grateful for meeting you in the lab. We are a team and will always be.

To all my other work colleagues: Leonor, Damian, Arturo, Simão, Ivan, Dasha – a wholehearted thank you for always being there to teach me and give me support. We had a lot of fun together, and I will forever cherish the moments we had – both at IGC and outside.

To all my friends from técnico – Alex, Bárbara, Carolina, Inês Carvalho, Inês Gonçalves, Inês Sá, Luís, Mariana, Peixinho, Sofia, Tiago – you know how much you mean to me. Sharing this experience with you made it 1000 times better. Thank you for all the support and caring from your side. To my deepest friend since kindergarten, Carolina Valentim, thank you for everything. You already know everything, sister. Your support and wisdom mean the world to me.

Mãe, não me vou estender muito no agradecimento, porque acredito que já saibas tudo. Obrigada por todo o apoio, compreensão e amor. Sem ti, tudo seria impossível. Leonor – és a melhor pessoa da minha vida. Não há ninguém que me ensine o que é a felicidade mais do que tu. Cada vez que chego a casa de um longo dia a trabalhar, sei que estarás à minha espera com o maior sorriso. Obrigada por tudo.

# Abstract

Super-resolution fluorescence microscopy techniques such as single-molecule localization microscopy (SMLM) enable imaging biological structures at nanoscale resolutions with unprecedented accuracy. However, analysing the large, complex datasets generated by SMLM experiments remains computationally demanding, hindering analysis throughput. In addition, the ability of SMLM localisation algorithms to accurately resolve high densities of fluorescent emitters represents a fundamental limitation. This thesis tackles these challenges through the development of optimised computational tools, analytical approaches and experimentation. First, a Python framework entitled NanoPyx is introduced to accelerate super-resolution image analysis on modern computing hardware. This is done by generating multiple implementations of a task and dynamically predicting the fastest for specific devices and input data. NanoPyx encompasses several methods that can be combined in a full super-resolution image analysis workflow. Next, simulations built within NanoPyx are used to systematically evaluate the counting accuracy of common SMLM localisation algorithms under varying known densities. Results reveal a density limit of $\sim$1 molecule/$\mu$m$^2$ for reliable quantification. Building on these simulations, an innovative competitive antibody binding strategy is devised to experimentally control labelling densities in a predictable linear fashion. By tuning the ratio of labelled and unlabelled antibodies targetting microtubules, the density-dependent performance of algorithms is characterised with real datasets. Together, these contributions in computational optimisation, simulation, and controlled experimentation aim to enhance the speed, reliability, and quantitative insights obtainable from single-molecule super-resolution imaging.

## Keywords

super-resolution | single-molecule localization microscopy | bioimage analysis | high-performance

# Resumo

Técnicas de microscopia de fluorescência de super-resolução, como microscopia de localização de moléculas únicas (SMLM), permitem a visualização de estruturas biológicas com resoluções na ordem de nanómetros com alta fidelidade. Contudo, analisar os extensos *datasets* adquiridos por estes métodos exige uma grande capacidade computacional. Adicionalmente, a capacidade dos algoritmos de localização de SMLM de resolver estruturas com elevadas densidades de fluoróforos com precisão representa uma limitação fundamental. Esta tese aborda estes desafios através do desenvolvimento de ferramentas computacionais otimizadas, abordagens analíticas e experimentais. Em primeiro lugar, é introduzida uma biblioteca em Python - NanoPyx - desenvolvida para acelerar computação para análise de imagem em super-resolução. Esta optimização é feita através da geração de múltiplas implementações de métodos e com a seleção dinâmica da mais rápida para cada *hardware* e *dataset*. Em segundo lugar, foram desenvolvidas simulações realistas de SMLM na biblioteca NanoPyx para sistematicamente avaliar a capacidade de quantificação de moléculas dos algoritmos de localização, revelando um limite de fidelidade de $\sim$1 molécula/$\mu$m$^2$. Em último lugar, foi desenvolvida uma estratégia experimental envolvendo competição de anticorpos para controlar linearmente a densidade de moléculas com fluoróforos. Variando o rácio de moléculas com e sem fluoróforos, a fidelidade dos algoritmos em contar moléculas pode ser caracterizada para diferentes condições de densidade, em experiênciais reais. O trabalho desenvolvido em otimização computacional, simulação e experiências contribui para um melhoramento na velocidade, fidelidade e caracterização quantitativa de técnicas de microscopia de super-resolução.

## Palavras Chave

super-resolução | microscopia de localização de moléculas únicas | análise de bioimagem

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **CPU** | Central Processing Unit |
| **CRLB** | Cramér-Rao Lower Bound |
| **eSRRF** | enhanced Super-Resolution Radial Fluctuations |
| **EM** | Electron Microscopy |
| **EMCCD** | Electron Multiplying Charge-Coupled Device |
| **FOV** | Field-of-View |
| **FRC** | Fourier Ring Correlation |
| **FHT** | Fast Hartley Transform |
| **FWHM** | Full Width at Half Maximum |
| **GIL** | Global Interpreter Lock |
| **GPU** | Graphics Processing Unit |
| **ISC** | Intersystem Crossing |
| **MLE** | Maximum likelihood estimation |
| **PALM** | Photoactivated Localization Microscopy |
| **PDF** | Probability Density Function |
| **PSF** | Point Spread Function |
| **RGC** | Radial Gradient Convergence |
| **ROI** | Region of Interest |
| **RSF** | Resolution Scaled Function |
| **RSE** | Resolution-Scaled Error |
| **RSP** | Resolution-Scaled Pearson Correlation |
| **RSF** | Resolution Scaled Function |
| **sCMOS** | Scientific Complementary Metal-Oxide-Semiconductor |

| | |
|---|---|
| **SMLM** | Single-Molecule Localization Microscopy |
| **SQUIRREL** | Super-Resolution Quantitative Image Rating and Reporting of Error Locations |
| **SRM** | Super-Resolution Microscopy |
| **SRRF** | Super-Resolution Radial Fluctuations |
| **STORM** | Stochastic Optical Reconstruction Microscopy |
| **SNR** | Signal-to-Noise Ratio |
| **TIRF** | Total Internal Reflection Fluorescence |
| **WLS** | Weighted Least Squares |

# 1 Introduction

## Contents

## 1.1 Theoretical Background

### 1.1.1 Fluorescence Principles

Fluorescence is the phenomenon from which molecules - fluorophores - emit photons upon the absorption of photons from a light source [1]. This process is the result of the transitions between different energy states of the molecule.

Typically, when a molecule is in its lowest energy state (ground state), its valence electrons are in a singlet electronic state ($S_0$). If the molecule absorbs photons with energy that is equal to or greater than the energy difference between two electronic states, there is a non-zero probability that an electron is excited to a higher energy state, a locally excited state ($S_{n,\, n \neq 0}$).

Once an electron is in an excited state, it can decay to the first excited state $S_1$ through a non-radiative process. From $S_1$, a series of energy release processes might occur depending on the molecular environment and structure. The electron can return to the ground state $S_0$ through a radiative decay, emitting fluorescence [2,3]. In addition, the electron might cross to an excited triplet state $T_1$, through Intersystem Crossing (ISC), as seen in Figure 1.1. This results in the formation of a longer-lived metastable dark state, as the reactions that follow the triplet state transition occur on much longer timescales (from $10^{-6}$s to 10s) in comparison to the fluorescence lifetime ( $10^{-9}$s). Canonically, the photons emitted by fluorescent molecules have longer wavelengths than the absorbed photons, due to an energy loss.

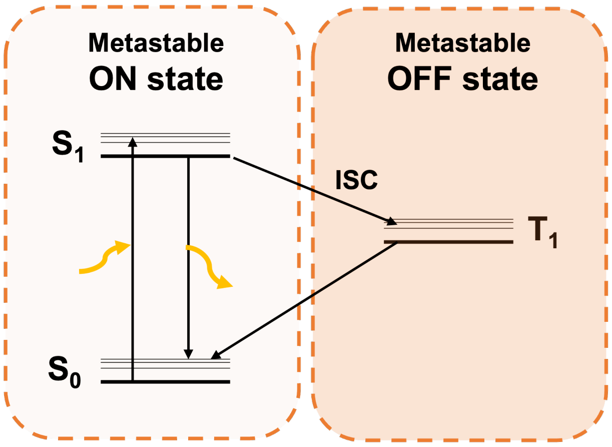


**Figure 1.1:** Simplified Perrin-Jablonski diagram [4] for molecular energy states and transitions. The electronic ground state $S_0$ and first excited singlet state and triplet states ($S_1$ and $T_1$) are represented. The radiative decay process of fluorescence ($S_1 \rightarrow S_0$) and inter-system crossing (ISC) ($S_1 \rightarrow T_1$) are also depicted. When ISC occurs, the electron transits to a metastable dark state.

Photoswitching is the stochastic phenomenon in which fluorophores switch between a metastable ON (bright) state, where they emit fluorescence, and a metastable OFF (dark) state, by a process commonly termed 'blinking' [5]. As explained above, the OFF state can be caused by ISC to a triplet state (see Figure 1.1). The rate at which photoswitching occurs can be controlled by adjusting the intensity of the illuminating light. Tipically, increasing the intensity of the light source results in a higher number of transitions between the ground and singlet state, increasing the likelihood of triplet state-induced dark states. Due to this behaviour, the photoswitching properties of fluorophores are regularly used in several fluorescence microscopy methods [6, 7].

### 1.1.2 Fluorescence Microscopy

Fluorescence microscopy is revolutionizing microscopic exploration by enabling scientists to observe and analyze intricate physiological processes within cells under both fixed and live cell conditions [8, 9]. This technology takes advantage of the use of fluorophores to precisely label distinct cellular components in a sample.

An epifluorescence microscope is mainly composed of a light source, an excitation filter, a dichroic mirror, an emission filter, an objective and a detector [2, 7], as illustrated in Figure 1.2.

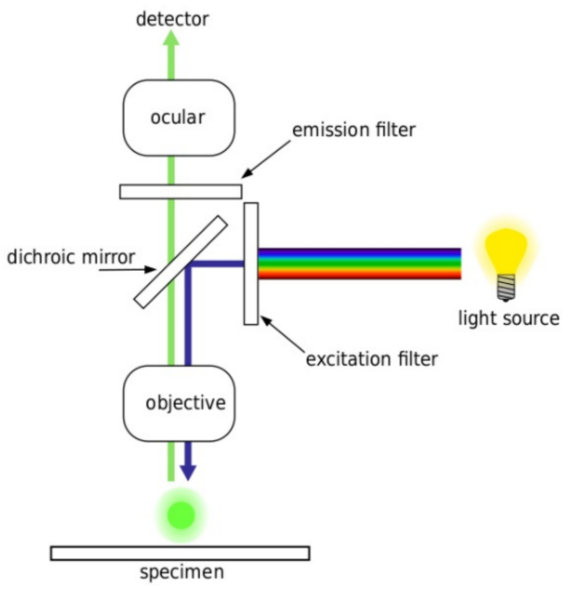


**Figure 1.2:** Illustration of an epi-fluorescence microscope. Adapted from Henry Mühlpfordt under CC BY-SA 3.0.

The light source used for excitation can be a laser, a lamp or a light-emitting diode (LED). The excitation filter selects specific wavelengths from the light source to be absorbed by the fluorophores in the sample. A dichroic mirror separates the excitation light that travels to the sample from the emitted light that travels from the sample to the detector. The mirror is placed at a 45-degree angle from the path of the light source, causing it to be reflected to the sample. Additionally, the dichroic mirror only allows the fluorescence emitted from the sample to pass through to the detector [7, 9]. These processes are possible due to the different wavelengths in the absorbed and emitted light. Since back-scattered excitation light can occasionally pass through the dichroic mirror, these microscopes are also equipped with an emission filter to select only the emitted fluorescence from the specimen.

The light emitted by the fluorescent molecules in the sample is detected by a camera, which converts the photons into an electric signal through the use of a photosensitive element. The camera sensor, an array of pixels, captures photons and generates electric charges within each pixel. The stored charges are then moved to a readout circuit, where they are converted into an electric signal. The resulting digital data is then processed to form an image. Different cameras have varying levels of quantum efficiency and noise. Quantum efficiency refers to the proportion of photons that are converted into electrons by the pixels in the sensor. Camera noise, on the other hand, can come from several sources, such as shot noise, readout noise, and optical background noise [10, 11].

Shot noise is caused by the random, statistical nature of the arrival of photons at the detector, and is not dependent on the type of camera used.

Readout noise is caused by electronic noise in the circuits used to read out the signal from the sensor. Two commonly used detectors that reduce readout noise are Electron Multiplying Charge-Coupled Device (EMCCD) cameras and Scientific Complementary Metal-Oxide-Semiconductor (sCMOS) cameras [1, 10].

Optical background noise is the unwanted light that is present in the environment and reaches the detector, and it is a common issue in fluorescence microscopy. To tackle this, Total Internal Reflection Fluorescence (TIRF) microscopy was developed to improve the signal-to-noise ratio [12]. This technique takes advantage of the optical phenomenon of total internal reflection, which occurs when light strikes the interface of materials with a higher refractive index at a sufficiently high angle (above the critical angle). In these conditions, the incident light gets entirely reflected to the first medium. In TIRF microscopy, the excitation light gets internally reflected in the interface between the coverglass and the liquid medium generating an electromagnetic field with the same frequency as the excitation light. This evanescent wave possesses an exponentially decaying intensity and is able to excite the fluorophores in the sample that are closer to the interface, up to 200 nm from the coverslip [13, 14]. This ultimately results in very low background fluorescence, making TIRF the preferred microscopy technique for visualising single emitters in a sample.

### 1.1.3 Overcoming the diffraction limit

Resolution can be defined as the minimum distance between two distinguishable signals. In light microscopy, the resolution can be improved with better design and manufacturing of the optical system until a fundamental physical limit is reached. This is called the Abbe's diffraction limit [7, 14, 15].

Diffraction causes light waves to "spread out" when they encounter a discrete slit or aperture. In an optical microscope, the lens will act as a circular aperture, which causes the emitted light to undergo diffraction, causing a diffraction pattern to be observed upon reaching the camera [2, 16]. This pattern is called the Point Spread Function (PSF), which is shaped by a series of concentric disks, also referred to as the "Airy Disk pattern", as seen in Figure 1.3. The Airy pattern of the PSF can be mathematically described as follows (equation 1.1).

$$PSF(r) = \left(\frac{2J_1(\alpha r)}{\alpha r}\right)^2, \alpha = \frac{2\pi NA}{\lambda} \tag{1.1}$$

Here, $PSF(r)$ represents the intensity of light at a distance $r$ from the center; $J_1$ is the Bessel function of the first kind, describing the oscillatory behaviour of the PSF [17]; $NA$ is the numerical aperture of the objective; and $\lambda$ is the wavelength of light used for imaging.

The resolving power of a microscope is related to the minimum distinguishable distance $d$ between the peaks of the PSFs [3,7] and is commonly quantified following Abbe's principle and Rayleigh's criteria, which define the limits of optical resolution [14], and is expressed through Equation 1.2.

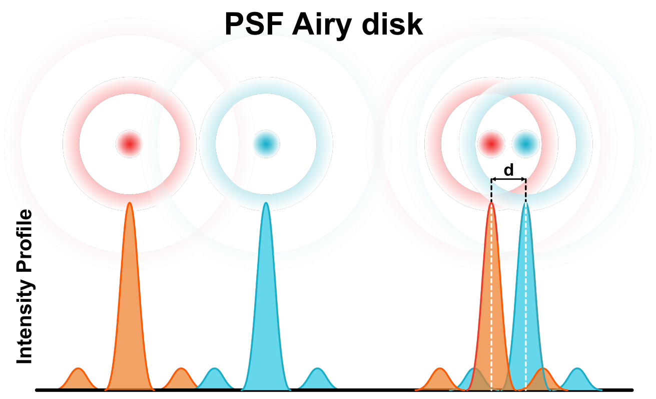


**Figure 1.3:** Airy disk pattern of a point spread function (top) and corresponding intensity profiles (bottom). The distance between the peaks of PSFs $d$ determines the resolution.

$$d = \frac{0.61\lambda}{NA} \tag{1.2}$$

Given that $NA$ typically takes values between 0.1 and 1.4, depending on the objective and presence of an immersion medium, the best achievable resolution of a fluorescent microscope is around 200 to 300 nm [2, 7].

This historical limit severely hampers the insight into biological architecture such as cellular structures and protein complexes, whose size ranges from a few to tens of nanometers.

This barrier forced the development of new microscopy methods. Electron Microscopy (EM), for instance, uses the wavelike properties of electrons, characterised by smaller wavelengths compared to light waves, to reveal structures within specially treated samples [16, 18].

EM can reach impressive resolutions below 0.2 nm, surpassing conventional optical microscopy by several orders of magnitude. Nevertheless, while offering exceptional resolution, electron microscopy is not currently compatible with live cell imaging, and its molecular specificity falls short of fluorescence microscopy techniques. This means that dynamic biological interactions and mechanisms cannot be observed via EM, which is a major limiting factor in biological research [16].

Super-Resolution Microscopy (SRM) is a set of optical techniques that emerged in the 2000s that can overcome the light's diffraction limit and resolve sub-cellular architecture down to tens of nanometers. This breakthrough not only retains the molecular specificity inherent to fluorescence microscopy but also elevates compatibility with live-cell imaging, when compared to other high-resolution microscopy methods such as EM [8]. Among the available SRM techniques, Single-Molecule Localization Microscopy (SMLM) can achieve one of the finest precisions [15].

### 1.1.4 Single-Molecule Localization Microscopy

SMLM is a set of imaging techniques that computationally detect and localise individual fluorescent molecules from diffraction-limited image sequences, generating a super-resolved sub-diffraction image [14]. SMLM methods include Photoactivated Localization Microscopy (PALM) [19,20] and Stochastic Optical Reconstruction Microscopy (STORM) [21]. Essentially, single-molecule localization is achieved by controlling the photoswitching rates of fluorophores in the sample. By forcing the majority of fluorescent molecules in an image frame into reversible OFF states, only a sparse subset of fluorophores are emitting light in that frame. If their PSFs are spatially separated at distances higher than the diffraction limit, i.e, if the PSFs don't overlap, it is possible to precisely detect and localise their position with computational algorithms. By repeating this cycle of photoactivations and readouts to a sufficiently high number of time frames, an image stack of thousands of images with individually resolvable molecules localisation is recorded. By stacking all the images together, a super-resolved sub-diffraction image is reconstructed [15, 22]. The framework of SMLM is illustrated in Figure 1.4.

SMLM has achieved impressive resolutions of sub-20 nanometers [14], well below the diffraction limit. An example of a diffraction-limited image and its reconstruction via SMLM can be seen in Figure 1.5.

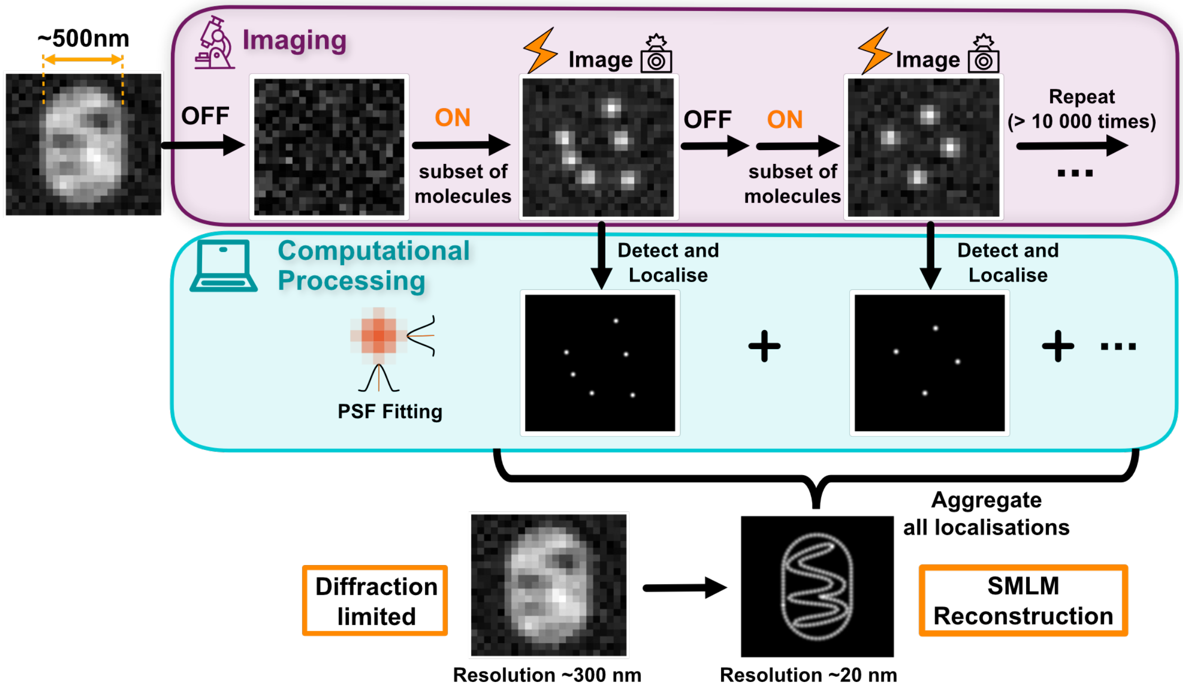


**Figure 1.4:** Single-molecule localization microscopy framework. SMLM imaging techniques promote stochastic blinking of fluorophores in a sample, with the consequent imaging of thousands of time frames with spatially separated emitters. For each obtained image frame, a computational framework performs detection and localisation of individual molecules, by fitting a PSF into the location of each fluorophore. Finally, by aggregating all the localisations from all the frames, a super-resolved image is reconstructed. The illustrated data is realistically simulated.

The forthcoming sections will provide a more detailed insight into the execution of typical SMLM experiments, encompassing the entire process from sample preparation to image analysis. Furthermore, an in-depth exploration of the present constraints and limitations inherent to SMLM will also be undertaken.

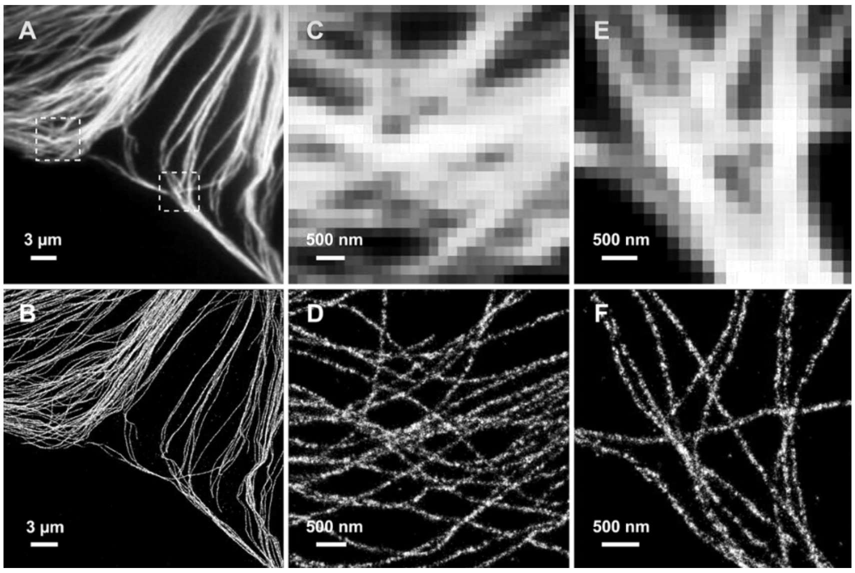


**Figure 1.5:** Diffraction-limited image of microtubules (A) and STORM image of the same area (B). (C, E) are the conventional and (D, F) are the STORM images corresponding to the cropped sections in (A). [23]

### Data acquisition

Imaging fixed samples for SMLM commonly requires immunolabelling. These methods involve the use of antibodies to specifically target and label the molecules of interest in a sample followed by acquisition using fluorescence microscopy. Immunolabelling is a technique that can be executed in two ways: direct or indirect. The direct method involves attaching a primary antibody to a fluorophore directly. In contrast, indirect immunolabelling does not involve the primary antibody carrying a fluorescent label. Instead, it binds to a secondary antibody that is bound to the fluorescent molecule [15, 24].

The interaction between an epitope and an antibody plays a crucial role in successful immunolabelling. The binding affinity between these two entities is instrumental in determining the specificity and accuracy of labelling. To ensure comprehensive labelling and minimise incompleteness in the observed structure, saturating antibody concentrations are employed. This strategy helps ensure that all available binding sites on the target epitope are occupied, enhancing the chances of successful labelling.

The probability of an epitope being labelled can be quantitatively described using the Hill-Langmuir equation [25], as follows.

$$P(labelled) = \frac{[Ab]^n}{[Ab]^n + K_d} \tag{1.3}$$

Whereas $[Ab]$ is the antibody concentration, $K_d$ is the dissociation constant and $n$ is the Hill coefficient.

In the realm of SMLM experiments, microtubules have emerged as a widely used biological structure for labelling and method validation due to their well-characterised filamentous nature. These slender, tube-like structures are integral components of a cell's cytoskeleton and possess a distinctive tubular structure with a diameter of approximately 25 nanometers. Microtubules consist of 13 protofilaments arranged in a circular configuration around a hollow core. Each protofilament generally comprises alternating $\alpha$-tubulin and $\beta$-tubulin monomers, and the width of each $\alpha$-tubulin/$\beta$-tubulin dimer is approximately 8 nm [26].

In microtubule immunolabelling, primary antibodies against either $\alpha$-tubulin or $\beta$-tubulin epitopes are employed to target the respective tubulin subunits. In the case of indirect immunolabelling, secondary antibodies are used to bind to the primary antibodies. Among the commonly used labels for SMLM studies, the photoswitchable dye Alexa Fluor 647 [27] stands out due to its exceptional photostability, brightness and inducible photoswitching properties.

A 2-dimensional representation of microtubule immunolabelling, considering the dimensions of the molecules involved, can be seen in Figure 1.6.

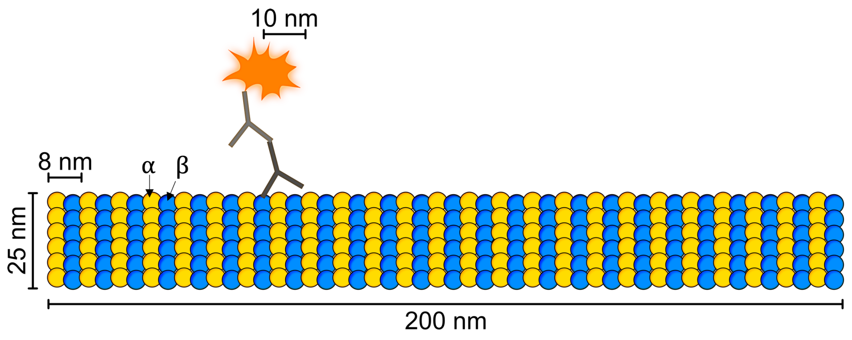


**Figure 1.6:** 2-dimensional representation of the microtubule structure with indirect immunolabelling in one protein. The yellow and blue monomers represent the $\alpha$-tubulin and $\beta$-tubulin, respectively. Indirect immunolabelling is represented for one of the monomers.

Following the completion of immunolabelling experiments targeting the structure of interest, the next step involves the imaging of the samples. To achieve high-quality super-resolution images of immunolabelled specimens, it's crucial to employ an appropriate imaging buffer that effectively regulates the photoswitching behavior of fluorophores. These buffers should include reducing agents, such as MEA or DTT, which facilitate the occurrence of the desired longer dark states in the fluorophores. Additionally, an oxygen scavenger system should be included to eliminate molecular oxygen from the solution [15,28]. $O_2$ exists in a triplet state in its ground state, and the chemical reactivity of fluorophores in triplet states can lead to the formation of singlet state oxygen—a highly reactive species that can induce photobleaching in fluorescent molecules. This photobleaching renders the molecules permanently unable to emit fluorescence. A commonly employed oxygen scavenger system uses a glucose oxidase and catalase enzymatic combination [15].

A crucial aspect of SMLM techniques revolves around the precise imaging conditions under which data is gathered. One frequently used approach for imaging fixed cells involves employing TIRF microscopy, due to its ability to provide high Signal-to-Noise Ratio (SNR)s, coupled with EMCCD or sCMOS cameras. Central to this acquisition methodology is the laser power used to excite the fluorophores in the sample. The laser power is adjusted to promote a desired photoswitching ratio, ensuring it is sufficiently high to induce photoswitching in the fluorescent molecules. The photoswitching ratio, expressed as $r = k_{off}/k_{on}$ [29], describes the connection between the rates of transition from the ON-state ($k_{on}$) to the OFF-state ($k_{off}$) of the fluorophores. A common target is a ratio of 1000, indicating the preference for fluorophores to spend 1000 times longer in the OFF-state than in the ON-state. To accomplish this, powerful laser settings are applied, typically falling within the range of 2 to 10 kW/cm$^2$ [30]. This excitation occurs with a relatively short exposure time of approximately 10 milliseconds per frame. This timing is chosen to align with the approximate duration of the fluorophores' ON-state [14]. This allows for efficient capture of the fluorescence emission while the fluorophores are in their active state. Subsequently, a substantial sequence of image frames is captured, often ranging from thousands to tens of thousands. This extensive frame acquisition guarantees that all emitters are imaged in their ON-state at least once. This accumulation of frames results in a diffraction-limited image stack where individual emitters are distributed sparsely across each frame. This controlled distribution is essential for accurate localisation of each emitter and subsequent super-resolution image reconstruction.

### Single-molecule detection and localisation

In SMLM, a computational framework is followed to process the acquired diffraction-limited image stack. The first step in generating a super-resolved reconstruction is to detect the most likely position of molecules in each frame. This is usually done by extracting pixels with the highest intensity values within a pixel neighbourhood equivalent to the size of the PSF via a local maximum filter algorithm [14, 15]. Theoretically, higher intensities in a pixel correspond to a higher number of photons detected.

Following the detection of an approximate emitter position in each frame, each molecule is precisely localised with localisation algorithms. Achieving sub-diffraction-level localisation for each molecule involves fitting a model of the PSF within the detected pixel window [17]. As mentioned in section 1.1.3, the PSF of a fluorescence microscope can be described by an Airy pattern. Nevertheless, studies have shown that a two-dimensional Gaussian shape serves as a suitable approximation for the fluorescent microscope's PSF [31], providing the benefit of simplifying subsequent analyses. The commonly employed 2D Gaussian function for approximating the PSF is defined as:

$$PSF(x,y) = \frac{1}{2\pi\sigma_0^2} \exp - \frac{((x-x_0)^2 + (y-y_0)^2)}{2\sigma_0^2} \tag{1.4}$$

Where $\sigma_0$ is the standard deviation of the Gaussian function, and $x_0$ and $y_0$ are the coordinates of the molecule's position. Given that the detectors used in acquisition focus the incoming light into a two-dimensional pixel array, it is necessary to modify the continuous distribution of the PSF to a discrete, pixelised profile [17, 32]. For this, each pixel is assumed to be rectangular and the expected number of photons in that pixel can be calculated by the integral of the Gaussian profile mentioned above 1.4 over the pixel area. This yields the following.

$$\mu_k(x,y) = I_0 \Delta E_x(x,y) \Delta E_y(x,y) + b_0 \tag{1.5}$$

Where $\mu_k$ is the expected number of photons in the pixel $k$, $I_0$ is the expected total number of photon counts, and $b_0$ is the background. $\Delta E_x$ and $\Delta E_y$ can be given by the following [17, 32].

$$\begin{aligned} \Delta E_x(x,y) &= \frac{1}{2}(erf(\frac{x - x_0 + \frac{1}{2}}{\sqrt{2}\sigma_0}) - erf(\frac{x - x_0 - \frac{1}{2}}{\sqrt{2}\sigma_0})) \\ \Delta E_y(x,y) &= \frac{1}{2}(erf(\frac{y - y_0 + \frac{1}{2}}{\sqrt{2}\sigma_0}) - erf(\frac{y - y_0 - \frac{1}{2}}{\sqrt{2}\sigma_0})) \end{aligned} \tag{1.6}$$

Where $x_0, y_0$ is the position of the emitter, and $erf$ is the error function [33]. Fitting can then be achieved with several algorithms, and the most commonly used are the Maximum likelihood estimation (MLE) and the Weighted Least Squares (WLS) [34–36].

MLE is a statistical method that aims to find the parameters that maximise the probability of the model matching the real values. MLE maximises the *likelihood* function (or its logarithm). Given that observed discrete counts (represented as $d_k$) can be assumed to draw a Poisson distribution, the likelihood can be represented by a Poisson process [35]. Thus, localising a specific emitter means finding which parameters maximise the following likelihood function.

$$\begin{aligned} \mathcal{L}(\theta|D) &= \prod_k \frac{\mu_k(x,y)^{dk} \exp^{-\mu_k}}{dk!} \\ \theta_{ML} &= \arg\max_\theta L(\mathbf{x}|\theta) \end{aligned} \tag{1.7}$$

The parameters $\theta$ correspond to the emitter positions $((x_i, y_i), i = 1...N)$ and the background $b_0$. To find these parameters, it is common to perform iterative approaches, such as the Newton-Raphson method [17].

WLS minimises the sum of the squared differences between the model and the actual pixel values, and it's parameter estimation is given by the following [37].

$$\theta_{WLS} = \arg\min_\theta \epsilon^2(\mathbf{x}|\theta) \tag{1.8}$$

Where $\epsilon$ is the difference between the model (prediction) and the real pixel values.

It is important to highlight that the imaging model ($\mu_k$) only considers a single emitter within the specified pixel area. This limitation can lead to challenges in precisely determining the emitter's position, particularly if other molecules are situated closely to or partially inside this pixel area. To tackle this, an enhancement of the model has been proposed, involving a technique known as multiemitter fitting [32]. In this approach, it is assumed that for a given pixel k, each emitter contributes independently to the expected photon counts. Therefore, in pixel k, the expected number of counts produced by N emitters can be calculated as follows.

$$\mu_k(x,y) = \sum_{i}^{N} I_0 \Delta E_{x_i}(x,y) \Delta E_{y_i}(x,y) + b_0 \tag{1.9}$$

A disadvantage of using multi-emitter fitting is that it is a computationally intensive method, limitting their practical use to process large biological datasets [38].

Several software packages that encapsulate these algorithms have been made available to researchers. One of the most widely adopted is ThunderSTORM [39], a plugin from ImageJ [40]/ FIJI [41].

With ThunderSTORM, users are required to provide their acquired diffraction-limited image stack and then select the desired reconstruction parameters. These parameters encompass critical aspects such as the magnification factor, pixel size, the choice of PSF model (with the default being the above mentioned integrated Gaussians), the detection method (typically defaulting to the local maxima approach), and the localisation algorithm – with options such as the MLE or WLS, with the additional possibility of multi-emitter fitting.

### Single-molecule localization microscopy limitations and requirements

Many factors influence the precision and accuracy of molecule localisation in SMLM [8, 15, 24]. Ultimately, these factors can be categorised into three domains related to the sample preparation, acquisition and analysis [42].

- Sample-related factors encompass attributes inherent to the biological sample and its preparation, which can significantly impact the acquisition and consequent accuracy of the reconstruction process [43]. These include the following.
  - Fixation: chemical cell fixation of biological samples can produce significant changes on cellular structures, subsequently impacting antibody binding. Optimised fixation protocols are needed to preserve the target structure as close to its native state as possible [44].
  - Labelling specificity: Nonspecific binding of antibodies conjugated with fluorophores can introduce background noise and unwanted signal contributions. Proper washing steps and control experiments help minimise this [45].
  - Artifact sources: autofluorescence or contaminants in the sample can contribute to unwanted

background signal. Additionally, some fixative agents introduce unwanted autofluorescence that needs to be quenched. Thorough washing and stringent controls help identify and troubleshoot these [46, 47].

  - Biological structure: naturally denser structures can lead to substantial overlap of PSFs, posing challenges in distinguishing individual molecules. Even within a single structure, localised densities can vary, as seen in overlapping structures, requiring careful considerations during the reconstruction process [48, 49].
  - Choice of fluorophores: only certain fluorophores exhibit the necessary photoswitching properties for SMLM, with specific switching kinetics and high photostability. Additionally, the fluorophore's quantum yield, i.e, the efficiency with which it emits photons upon excitation, also plays a vital role. Moreover, some fluorescent molecules are more prone to photobleaching than others, which affects the total photon yield over the acquisition time [50, 51]. Commonly used fluorophores include Alexa Fluor 647 [27], Alexa Fluor 568 [52], CF 680 [53] and Atto 488 [54] [15].
  - Blinking buffer: mounting the sample with an appropriate blinking buffer is crucial to ensure the desired photoswitching behaviour of fluorophores. As previously stated, this buffer should contain a reducing agent to promote longer dark states and an oxygen scavenger system to eliminate molecular oxygen from the solution [55].

- Acquisition-related factors include the hardware used to acquire the data and the physical limits imposed by the imaging system. These encompass the following.

  - Camera characteristics: the pixel size should be $\leq$ 1/2 the width of the PSF to avoid undersampling errors (i.e, pixel sizes should be 100-150 nm); the quantum efficiency should be high to maximise the number of detected photons ($\geq$ 70%); the readout noise should be low (less than 1 electron per pixel); the frame rate should be high to capture the fast blinking dynamics of the fluorophores ($\geq$ 50 fps) [14, 56].
  - Optics: The numerical aperture of the objective influences the PSF width and shape. Higher numerical apertures yield smaller, more symmetric PSFs, which aid precision [57].
  - Excitation intensity: as previously mentioned, high laser powers (2-10 kW/cm$^2$) are required to promote photoswitching, to spatially separate the emitters measured PSFs [30]. Yet, excessively high laser power leads to photobleaching, constraining the overall photon yield over time.
  - Stage drift: Sample drift during acquisition causes blurring and localisation errors. Post-processing drift correction techniques help compensate for this [58].

- Out-of-focus light: Background fluorescence from above/below focal plane contributes noise and degrades precision. Optical sectioning techniques like TIRF mitigate this [13].

  In fact, given the nature of the acquired signal, there is a fundamental limit as to how precisely each emitter can be localised. Assuming that the dominant source of noise in the acquired signal is photon shot noise (see section 1.1.2), the following expression shows the uncertainty of the location of an emitter [14, 59].

$$\Delta x = \sqrt{\frac{\sigma^2 + a^2/12}{N}} \tag{1.10}$$

  Where $\sigma$ is the standard deviation of the PSF, $a$ is the pixel size and $N$ is the number of photons collected by the detector. This means that the higher the number of photons, the better the precision (lower uncertainty). This quantitative notion is defined as the Cramér-Rao Lower Bound (CRLB) limit [60], and it is widely used to compare localisation algorithms, given they should obey this theoretical limitation.

• Analysis-related factors entail the computational framework used to process the acquired data.

  - Localisation algorithms. Common algorithms like MLE and WLS differ in how they fit a model PSF to determine the molecule's position. MLE generally provides highest precision. However, all localisation algorithms have fundamental limits in accuracy and precision when dealing with overlapping PSFs due to high local emitter densities high local emitter densities. In such cases, algorithms cannot reliably distinguish nearby emitters, leading to undercounting of molecules and localisation errors [58, 61, 62]. This sets an upper density limit for reliable quantification and super-resolution imaging. Approaches like multi-emitter fitting aim to push this density limit higher, but the problem remains a key challenge in SMLM.
  - Post-processing: drift correction, registration, and other post-processing steps can help improve precision and accuracy [58].

The various factors taken into account in SMLM demonstrate the extensive complexities involved in hardware design and optimisation to achieve super-resolution reconstructions from single molecule signals with high precision and accuracy. State-of-the-art cameras, objectives, lasers, and mechanical stability are essential to maximise photon collection and minimise noise and artifacts. At the same time, consideration of the sample characteristics, labeling strategies, acquisition modes, and analysis algorithms all aim to promote low densities of active fluorophores in each frame. This mitigates issues with overlapping PSFs that fundamentally limit localisation precision and accuracy at high densities.

To tackle this, alternative computational techniques to generate super-resolution images have emerged, attempting to deal with higher-density data.

### 1.1.5 Fluctuation-based Super-Resolution algorithms

The dependence of SMLM techniques on the existence of long dark states by fluorophores motivated the development of computational methods based on statistically independent blinking patterns of fluorophores [63]. The developed algorithms exploit the small intensity fluctuations of the fluorescence signal in the time-series. Therefore, contrarily to SMLM, these techniques reconstruct a super-resolved image directly from the image data, not from precisely detecting and localising each fluorophore's position [2, 3].

One of the most widely adopted fluctuation-based super-resolution algorithm is Super-Resolution Radial Fluctuations (SRRF) [64]. SRRF takes advantage of the fact that acquired PSFs exhibit approximate radial symmetry in the image plane. SRRF magnifies each pixel of a diffraction-limited image frame into subpixels (via a Catmull-Rom interpolation [65]) and assigns a *radiality* value to each subpixel, representing a probability of an emitter being present in that location. This value is a measurement of the intensity gradient convergence [64]. Theoretically, the intensity gradient in a given position will point towards the centre of the fluorophore originating the PSF [66]. In addition to the spatial analysis, performed over all the frames of the diffraction-limited image stack, the SRRF algorithm performs temporal analysis of the acquired stack. This is done by applying temporal correlations between frames: radiality peaks caused by noise are not correlated over time (causing their temporal correlation to be close to zero), and the highest degree of correlation in time is located at the centre of radiality peaks produced by the real emitters [64]. An illustration of the SRRF algorithm can be seen in Figure 1.7.

Experimentally, SRRF showed to achieve resolutions down to 40 nm, and is currently one of the most widely used non-localisation-based methods by the super-resolution community. It is available to users as a plugin for ImageJ [40]/ FIJI [41].

More recently, an enhancement of the SRRF algorithm, named enhanced Super-Resolution Radial Fluctuations (eSRRF) [67] was developed, which improves the image fidelity and resolution of the previously described algorithm SRRF. There are some key differences in the spatial analysis of eSRRF, when compared to the original SRRF. To subpixelise the original images, eSRRF uses a Fast Hartley Transform (FHT) [68], as oposed to a Catmull-Rom interpolation, which was proven to minimise macro-pixel patterning artifacts in the reconstruction. Most importantly, the mathematical model used to calculate the *radiliaty* value, now referred to as Radial Gradient Convergence (RGC), was improved: it now considers the weighted influence of the intensities of all subpixels surrounding each potential emitter, in the continuos space. This allows a better estimation of the environment around the pixel of interest, and ultimately permits enhanced image fidelity and resolutions. The performed spatial analysis outputs several RGC maps, one for each frame of the acquired image stack. These maps are then correlated in time to generate a super-resolved image.

A comprehensive mathematical description of the eSRRF algorithm is provided in Appendix A.1.

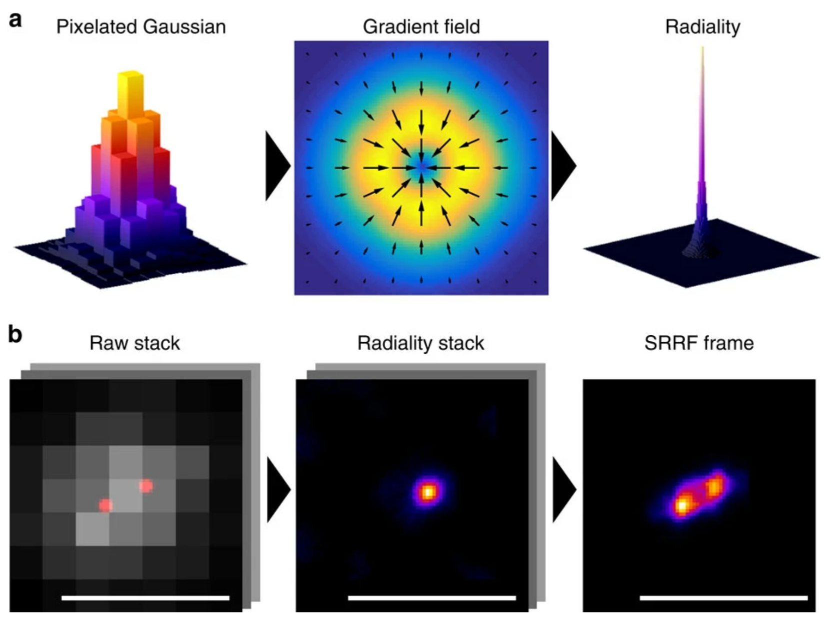


**Figure 1.7:** **a)** Left: illustration of the obtained diffraction-limited PSF in 3D. Middle: surface plot of the gradient field of a PSF. The arrows indicate the direction of the gradient vectors, which are pointing towards the centre of the gradient field due to its radial symmetry. Right: illustration of the obtained radiality (degree of convergence of the intensity gradient vectors) in 3D. **b)** Left: simulation of a diffraction-limited image stack of 100 frames with two emitting fluorophores separated by a distance of 135 nm, smaller than the diffraction limit. The true positions of the fluorophores are indicated in red. Middle: obtained radiality maps for each frame after applying the SRRF algorithm. Right: output SRRF image acquired by temporal analysis of the stack of radiality maps. Scale bar: 500 nm [64].

### 1.1.6 Image analysis tools for super-resolution microscopy

Most super-resolution approaches rely on analytical steps to process the acquired data. These encompass image registration techniques, resolution enhancement - accomplished through SMLM algorithms or fluctuation-based approaches like SRRF and eSRRF - and the quantification of features of the acquired data such as quality and resolution.

Image registration techniques encompass two essential components: drift correction and channel registration [69]. Drift correction entails aligning the image stack to compensate for sample drift during acquisition. This is frequently accomplished through cross-correlations between frames. Meanwhile, channel registration focuses on aligning distinct channels in multi-channel acquisitions, to mitigate chromatic aberrations [70].

Performing quantitative quality control of the generated data is essential to both validate and compare different methods. One widely used method to assess the quality of super-resolved images is Super-Resolution Quantitative Image Rating and Reporting of Error Locations (SQUIRREL) [71]. SQUIRREL generates a quantitative error map of super-resolution defects by comparing the original diffraction-limited data to a simulated diffraction-limited version of the super-resolved reconstruction.

The super-resolved image is convolved with a Resolution Scaled Function (RSF), optimised to match the microscope's PSF, to generate the simulated diffraction-limited image for comparison. The pixel-wise error between this and the original diffraction-limited data produces a map of defects in the super-resolution image. In addition, SQUIRREL calculates two quantitative global image metrics: the Resolution-Scaled Error (RSE), representing the root-mean-square error between the reference and resolution-scaled image; and the Resolution-Scaled Pearson Correlation (RSP) coefficient. An example of an error map obtained can be seen in Figure 1.8.

Besides assessing the quality of a super-resolved image, it is essential to calculate the achieved resolution. A popular method for quantifying the resolution of a reconstruction is with Fourier Ring Correlation (FRC) [72]. It works by splitting the localisation data into two subsets, generating separate images from each subset, then calculating the correlation between the Fourier transforms of these images along concentric rings of increasing spatial frequency. The FRC curve plots this correlation as a function of frequency. The resolution is defined as the inverse of the frequency at which the FRC correlation falls below a chosen threshold. This transition indicates where the subset images lose correlation, defining the achievable resolution limit. An example of an obtained FRC maps can be seen in Figure 1.8.

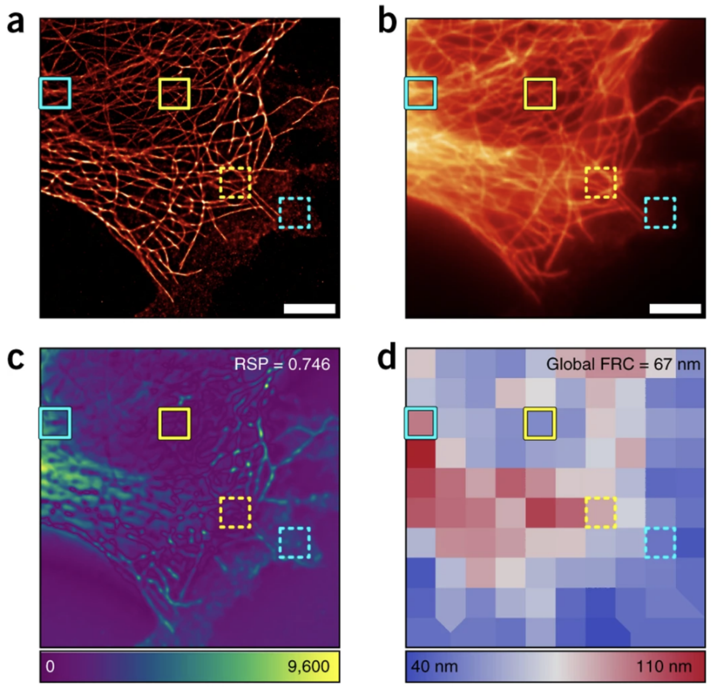


**Figure 1.8:** **(a)** SMLM reconstructed image (microtubule immunolabelling with Alexa Fluor-647). **(b)** Correspondent TIRF image. **(c)** Error map of **(a)** using **(b)** as a reference. **(d)** Local map of FRC values for the super-resolution image **(a)**. [71]

An alternative approach to calculate image resolution is with Decorrelation analysis [73]. This method directly computes the spatial correlation between the subset localizations as a function of distance. Here the resolution is defined as the distance where this spatial correlation plot decays below a set threshold value. By quantifying the distance or frequency at which random subsets decorrelate, both FRC and decorrelation analysis provide quantitative metrics to characterize super-resolution image resolution. However, the subjective choice of threshold impacts the absolute resolution value reported.

Most of the methods mentioned above find their implementation within the framework of NanoJ [69], a widely adopted ecosystem of ImageJ plugins for super-resolution microscopy. NanoJ includes NanoJ-Core, implementing image registration methods and acting as a library to be used by the other plugins; NanoJ-SRRF and -eSRRF, incorporating the methods SRRF and eSRRF; NanoJ-SQUIRREL, with the SQUIRREL Error map and the resolution metric FRC for quality control.

These techniques involve processing large image stacks and iterative optimisations, which is very computationally demanding. To accelerate processing, NanoJ leverages OpenCL [74] parallel programming. With OpenCL, algorithms are written in a C-like language with additional commands to launch parallel compute kernels. OpenCL handles distributing the workload across available compute units, enabling the harnessing of powerful parallel hardware like Graphics Processing Unit (GPU)s for large performance gains.

### Python for bioimage analysis workflows

In recent years, Python has become the favored programming language within the microscopy image analysis community. One of the reasons for it is the abundance of specialised libraries and frameworks designed for bioimage analysis. Being a high-level language, Python prioritises readability and user-friendliness, liberating developers from concerns about low-level memory management and machine-specific optimisations. As a result, developers can fully concentrate on implementing their desired algorithms. The versatility of Python is another advantage, as it can be seamlessly executed across various platforms, including diverse operating systems and hardware configurations. The surge in deep learning techniques has further propelled Python's popularity, particularly in the realm of bioimage analysis. Researchers now frequently implement cutting-edge deep learning methods in Python, leveraging the support of specific libraries tailored to this domain. The availability of these tools has enabled groundbreaking advancements in bioimage analysis, strenghtening Python's position as an ideal choice for creating advanced frameworks in this area.

However, standalone Python is an inherently slow programming language. Two of the factors contributing to this are 1) the fact that Python is an interpreted language and 2) the Global Interpreter Lock (GIL). Being an interpreted language means that Python is not directly executed by the computer, but rather by an interpreter. This makes Python code slower than compiled languages, such as C or

C++. In addition, Python has a GIL, which prevents the code to be ran on more than one thread (a unit of code execution) of the computer Central Processing Unit (CPU). This means that pure Python code cannot be executed in parallel, which is a problem for computationally intensive tasks.

### Achieving Computational Acceleration in Python

There are several methods to accelerate Python code by bypassing the GIL and compilation to machine code. Three popular approaches are Cython, PyOpenCL and Numba:

- Cython [75] is a static compiler that converts Python code into optimised C/C++ code that can be compiled into a Python extension module. It provides Python-like syntax while supporting calling C functions and declaring C types. Cython code runs significantly faster than Python because it bypasses the GIL to allow multi-threading (with the use of `prange`) and performs low-level optimisations like loop unrolling. One limitation is that Cython requires explicit type declarations, which removes some of Python's dynamism.
- PyOpenCL [76] allows Python code to execute parallel computations on GPUs through the OpenCL [74] framework. Computational tasks are offloaded to the GPU, which has thousands of tiny processing cores suited for data-parallel operations. PyOpenCL translates Python functions into OpenCL kernels that run efficiently on GPUs. This offers massive parallelism and speedup compared to Python limited by single-CPU core execution. Despite PyOpenCL not requiring a physical GPU to run, the best and easiest performance improvement requires one, which can be a limiting factor for some users.
- Numba [77] is a just-in-time (JIT) compiler that converts Python functions into optimised machine code. It is designed to accelerate numerical workloads using NumPy arrays and math operations. Numba-compiled code avoids interpreter overhead (i.e., excess computation time) and leverages vectorisation, loop-unrolling and parallel execution on multicore CPUs. One disadvantage of Numba is that it has compilation overhead on first run.

## 1.2 Problem Statement

Super-resolution microscopy techniques have become vital tools for biological studies. However, analysing the large, complex datasets acquired by these methodologies poses major challenges. Existing image analysis methods suffer from computational performance bottlenecks that hinder processing of massive modern datasets. Moreover, SMLM techniques face fundamental limitations in accurately localising and quantifying high densities of emitters. At high densities, overlapping PSFs lead to undercounting artifacts. However, systematically characterising this density-dependent behaviour is inherently difficult in real experimental conditions due to challenges in precisely controlling labelling densities.

Together, these issues of computational scalability and characterisation of localisation errors under high density imaging conditions represent problems this thesis seeks to address through the development of optimised tools and analytical approaches.

## 1.3 Project Objectives

The scope of this project falls in the development of optimised computational tools and systematic characterisation of SMLM localisation algorithms on both synthetic and real data. The thesis is subdivided into three chapters, each addressing a specific aim (see Figure 1.9).

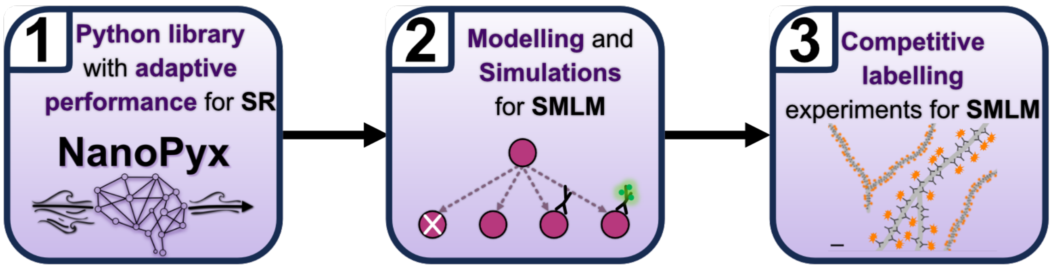

1
Python library with adaptive performance for SR
NanoPyx
2
Modelling and Simulations for SMLM
3
Competitive labelling experiments for SMLM

**Figure 1.9:** Project Aims. These include **1)**, the development of an adaptive library for super-resolution image processing; **2)** the implementation of simulations and modelling to assess SMLM algorithms and **3)** the development of a new experimental framework to evaluate algorithm performance.

**Aim 1:** Develop a high-performance and adaptive image analysis library to accelerate processing of large super-resolution datasets, incorporating super-resolution tools.

**Aim 2:** Implement simulations into the previous framework to systematically evaluate the performance of SMLM localisation algorithms under varying density conditions.

**Aim 3:** Develop experimental strategies to control molecular density and use these to evaluate real-world algorithm performance across a range of densities.

Together these computational and analytical advancements aim to enhance the accessibility, reliability, and biological insights obtainable from single-molecule super-resolution imaging.

# 2

# NanoPyx: adaptive bioimage analysis

## Contents

This chapter focuses on the development of NanoPyx, a Python library specifically designed for super-resolution image analysis with high-speed processing.

## Contributions

The NanoPyx framework was developed by the author of this thesis, Inês Martins Cunha, along with Bruno Saraiva and António Brito, with contributions from Robert Haase and Guillaume Jacquemet. In specific, the core of the NanoPyx framework - the Liquid Engine - was co-developed by Inês Cunha, Bruno Saraiva and António Brito. The image registration methods, including drift correction and channel registration, were implemented mainly by Bruno Saraiva; The implementation of super-resolution generation methods, including SRRF and eSRRF, was led by Inês Cunha; The quality control metric (Error Map of SQUIRREL) was the responsibility of Inês Cunha. The other quality control metrics, including FRC and Decorrelation analysis, were implemented by Bruno Saraiva. The single-molecule simulations were the responsibility of Inês Cunha, and described in detail in the next chapter. Raquel Portela prepared samples to showcase the NanoPyx framework, and Inês Cunha acquired and processed the data. Gautier Follain carried out additional experiments and acquisitions. Gautier Follain, Robert Haase, Pedro Pereira and Guillaume Jacquemet provided feedback, testing and guidance. Inês Cunha, Bruno Saraiva and António Brito further tested, documented and created user interfaces for NanoPyx. The entire endeavor was supervised by Ricardo Henriques.

## Code availability

The NanoPyx Python library can be found in the Github repository: https://github.com/HenriquesLab/NanoPyx.

## Manuscript Publication

The developed work has been published as a Preprint in https://doi.org/10.1101/2023.08.13.553080 [78]. In the first two weeks of becoming available, it reached over 700 downloads and is on the top 3% of all research outputs scored by Altmetric.

## 2.1 Introduction

Super-resolution microscopy has revolutionised cell biology by enabling fluorescence imaging at an unprecedented resolution [79–82]. However, the data collected from super-resolution experiments requires specific analytical methods, such as drift correction, channel alignment, resolution enhancement, and quantifying data quality and resolution. To execute these tasks effectively, various open-source image analysis software tools are commonly employed. Notably, platforms such as ImageJ [40] or Fiji [41] play a pivotal role, providing a wide range of tools such as ThunderSTORM [83], Picasso [84], FairSIM [85], Fourier Ring Correlations (FRC) [72], and Decorrelation Analysis [73].

However, as the adoption of super-resolution microscopy continues to expand, the tools used for image analysis become increasingly intricate. At the same time, the acquired datasets for these techniques are growing in terms of number, size, and complexity. The computational performance of existing methodologies has become a significant bottleneck, hindering the high-throughput analysis of these extensive datasets. This highlights the need for a shift towards a more performance-centric approach for super-resolution image analysis.

Furthermore, as explained in 1.1.6, there are multiple ways to obtain computational acceleration for an algorithm. In fact, an algorithm can potentially be ran in the computer CPU, either in a single core or with multiple cores, or in the GPU. Choosing the fastest implementation to run a specific computational task is not trivial. The reason for this is that, in theory, the fastest way of running a method will depend on several factors, such as 1) the method itself (how computationally intensive it is), 2) the specific device used to run it (the number of cores in the CPU, the type of GPU, etc) and 3) the data used as input (the size of the image(s)). In addition, it is important to note that some of the implementations might not be available to a user.

To tackle this, NanoPyx was developed: a high-performance and adaptive bioimage analysis framework. At the core of NanoPyx is the Liquid Engine, an agent-based machine-learning system that predicts acceleration strategies for image analysis tasks. The Liquid Engine uses multiple implementations of the same algorithm to perform a specific task. Although these implementations provide numerically identical outputs for the same input, their computational performance differs by exploiting different computational strategies. The Liquid Engine can then predict the optimal combination of implementations based on the user's specific hardware device and input datasets.

NanoPyx offers a variety of bioimage analysis methods, specifically tailored to super-resolution microscopy. These include categories available from the widely used NanoJ [69] plugin, such as Image Registration (drift correction and channel alignment [69]), Super-Resolution Generation (SRRF [64] and eSRRF [67]), and Quality Control (Error Map [71], FRC [72], and a new implementation of Decorrelation Analysis [73]) (see Figure 2.1). Bringing the adaptability of the Liquid Engine into these methods allows NanoPyx to overcome many limitations of NanoJ and other modern bioimage analysis packages. By providing a flexible framework accessibility of both new and old image analysis pipelines is assured, regardless of the user hardware, without sacrificing performance. Furthermore, this flexibility can be leveraged and used in conjunction with other Python libraries and tools. This is particularly valuable as many methods increasingly rely on Python-based deep learning techniques, making the use of current analysis frameworks (such as NanoJ) prohibitive in specific scenarios.

To make NanoPyx accessible to users with different levels of coding expertise, it is offered through three separate platforms - as a Python library for developers with the skills to create their workflow scripts, as Jupyter Notebooks [86] that can be executed either on a local machine or on a cloud-based service like Google Colaboratory, and as a plugin for napari [87], a Python based image viewer, for users without programming experience. By distributing NanoPyx in this manner, the needs of a wider audience are catered, ensuring users of varying coding expertise have easy access and can effectively utilise NanoPyx for their bioimage analysis needs.

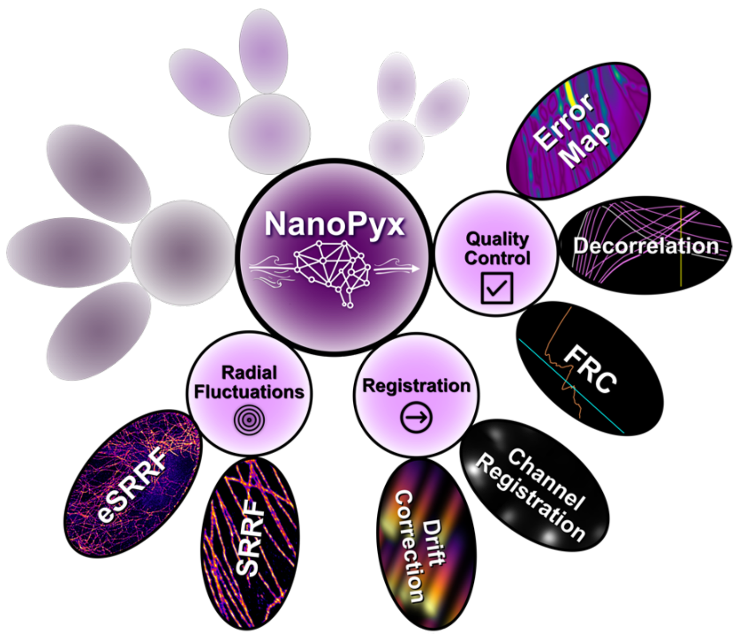


**Figure 2.1: Schematic representation of the NanoPyx framework.** NanoPyx is a Python framework for super-resolution microscopy images. It uses the Liquid Engine for self-tuning high performance. Currently, NanoPyx offers methods for Image Registration [69], Radial Fluctuations [88], and Quality Control [71] categories.

## 2.2 Methodology

This section covers specific methodology used to develop and test NanoPyx. It includes how the data used to showcase NanoPyx was acquired; how the Liquid Engine's agent was implemented; and which computational setups and dataset sizes were used to run benchmarks and comparisons.

### Mammalian cell culture

A549 cells (The European Collection of Authenticated Cell Cultures (ECACC)) were cultured in phenol red-free high-glucose, L-Glutamine containing Dulbecco's modified Eagle's medium (DMEM; Thermo Fisher Scientific) supplemented with 10% (v/v) fetal bovine serum (FBS; Sigma), 1% (v/v) penicillin/streptomycin (Thermo Fisher Scientific) at 37 ℃ in a 5% CO2 incubator.

### Sample preparation for microscopy

A549 cells were seeded on a glass bottom $\mu$-slide 8 well (ibidi) at a 0.05 - 0.1 x 106 cells/cm2 density. After 24 h incubation at 37 ℃ in a 5% CO2 incubator, cells were washed once using phosphate-buffer saline (PBS) and fixed for 20 min at 23 ℃ using 4 % paraformaldehyde (PFA, in PBS). After fixation, cells were washed three times using PBS (5 min each time), quenched for 10 min using a solution of 300 mM Glycine (in PBS), and permeabilised using a solution of 0.2% Triton-X (in PBS) for 20 min at 23 ℃. After three washes (5 min each) in washing buffer (0.05% Tween 20 in PBS), cells were blocked for 30 min in blocking buffer (5% BSA, 0.05% Tween-20 in PBS). Samples were then incubated with a mix of anti-$\alpha$-tubulin (1 $\mu$g/mL of clone DM1A, Sigma; 2 $\mu$g/mL of clone 10D8, Biolegend; 2 $\mu$g/mL of clone AA10, Biolegend) and anti-septin 7 (1 $\mu$g/mL of #18991, IBL) antibodies for 16 h at 4 ℃ in blocking buffer. After three washes (5 min each) using the washing buffer, cells were incubated with an Alexa Fluor™ 647 conjugated goat anti-mouse IgG and an Alexa Fluor™ 555 conjugated goat anti-rabbit IgG (6 $\mu$g/mL in blocking buffer) for 1 h at 23 ℃. Cells were then washed thrice (5 min each) in washing buffer and once in 1X PBS for 10 min. Finally, cells were mounted with a GLOX-MEA buffer (50 mM Tris, 10 mM NaCl, pH 8.0, supplemented with 50 mM MEA, 10% [w/v] glucose, 0.5 mg/ml glucose oxidase, and 40 $\mu$g/ml catalase).

### Image acquisition

Data acquisition was performed with the Nanoimager microscope (Oxford Nanoimaging; ONI) equipped with a 100 x oil-immersion objective (Olympus 100x NA 1.45) Imaging was performed using 405-nm, 488-nm, and 640-nm lasers for Hoechst-33342, AlexaFluor555 and AlexaFluor647 excitation, respectively. Fluorescence was detected using a sCMOS camera (ORCA Flash, 16 bit). For channel 0, a dichroic filter with the bands of 498-551 nm and 576-620 nm was used; for channel 1, a 665-705 nm dichroic filter was used. The sequential multicolor acquisition was performed for AlexaFluor647, AlexaFluor555 and Hoechst-33342. Using an EPI-fluorescence illumination, a pulse of high laser power (90%) of the 640-nm laser was used, and 10 000 frames were immediately acquired. Then, the sample was excited with the 488-nm laser (13.7% laser power), whereas 500 frames were acquired, followed by the 405-nm laser excitation (40% laser power), with an acquisition of another 500 frames. For all acquisitions, an exposure time of 10 ms was used.

## Liquid Engine's agent

Run times of methods implemented in NanoPyx through the Liquid Engine are locally stored on users' computers and are associated with the used hardware. For OpenCL implementations, the agent also stores an identification of the device and is capable of detecting hardware changes. Whenever a method is run through the Liquid Engine, the overseeing agent splits the 50 most recent recorded runtimes into 2 halves: one with the 25 fastest run times (fast average, $FastAvg$) and one with the 25 lowest (slow average, $SlowAvg$). Then it calculates the average of the 25 fastest run times for each implementation and selects the implementation with the lowest average runtime. Once the method finishes running, the agent checks whether there was a delay ($Delay$), which is defined by the last runtime being higher than the previously recorded average runtime of the fastest runs ($Expected$) plus two times the standard deviation ($Std$) of the fastest runs (Equation 2.1). If a delay is detected (Supplementary Figure A.4), the agent will also calculate the delay factor ($DelayFactor$, Equation 2.2) and will activate a probabilistic approach that stochastically selects which method to run. This is performed by using a Logistic Regression model to calculate the probability of the delay ($P_{delay}$) being present on the next run and adjusting the expected runtime of the delayed implementation according to Equation 2.3, while still using the fast average for all non-delayed implementations. Then, the agent picks which implementation to use based on probabilities assigned to each implementation ($P_{selecting}$) using 1 over the squared normalised expected runtime (Equation 2.4). This stochastic approach ensures that the agent will still run the delayed implementation from time to time to check whether that delay is still present. The agent decides that the delay is over ($Delay_{end}$) once the last runtime becomes smaller than the slow average minus the standard deviation of the slowest runs or higher than the fast average plus the standard deviation of the fastest runs (Equation 2.5). Once the delay is over, the agent will go back to selecting which implementation to use based only on the fast average of each implementation (Supplementary Figure 2.8).

$$Delay = Measured > (Expected + 2 \times Std) \tag{2.1}$$

$$DelayFactor = \frac{Measured}{Expected} \tag{2.2}$$

$$Adjusted = FastAvg \times (1 - P_{delay}) + FastAvg \times DelayFactor \times P_{delay} \tag{2.3}$$

$$P_{selecting} = \frac{1}{(ExpectedRuntime)^2} \tag{2.4}$$

$$Delay_{end} = Measured < (SlowAvg - Std \vee > FastAvg + Std) \tag{2.5}$$

## Run times Benchmark

For the laptop benchmarks a MacBook Air M1 Pro with 16Gb of RAM and a 512Gb SSD was used. For the professional workstation, a custom-made desktop computer was used containing an Intel i9-13900K, a NVIDIA RTX 4090 with 24Gb of dedicated video memory, a 1TB SSD and 128Gb of DDR5 RAM. The first benchmark performed (Figure 2.2 and Supplementary Figure A.2) was a 5 times up sampling of the input data, using a catmull-rom [89] interpolator. Benchmarks were performed on 3 different input images with shapes 10x10x10, 10, 10x300x300 and 500x300x300 (time-points, height, width). The second benchmarks (Supplementary Figure 2-4) were 2D convolutions using a kernel filled with 1s with varying sizes (1, 5, 9, 13, 17, 21) on images with varying size (100, 500, 1000, 2500, 5000, 7500, 10000, 15000 or 20000 pixels for both dimensions).

## NanoPyx comparison with NanoJ

Run times of eSRRF image processing, both in NanoPyx and NanoJ, were measured using a MacBook Air M1 with 16Gb of RAM and a 512Gb SSD. The parameters used for the analysis where the same for both NanoPyx and NanoJ: magnification – 5; radius – 1.5; sensitivity – 2; number of frames for eSRRF – 1. The input image was a stack with 283 by 283 pixels and 10 000 frames. For the final image output an average reconstruction was performed.

To assess the implementation of eSRRF into NanoPyx, it was used the publicly available dataset [90], an image stack with a Field-of-View (FOV) of 128 by 128 pixels and 500 frames. All analysis were performed with a MacBook Pro M1 with 16Gb of RAM and a 512Gb SSD.

## 2.3 Results

This section provides details into the development of NanoPyx, including its core platform (the Liquid Engine) and the methods integrated into NanoPyx. The implementation of the eSRRF method was explained in detail in Appendix A.2.2.

### 2.3.1 The Liquid Engine

At the core of NanoPyx resides the Liquid Engine. The Liquid Engine is a system that is able to predict the fastest implementations for image analysis tasks, tailored to each user device and input data.

It was found that identifying the most efficient implementation for a computational task significantly relies on the input data and available hardware, as depicted in Figure 2.2.

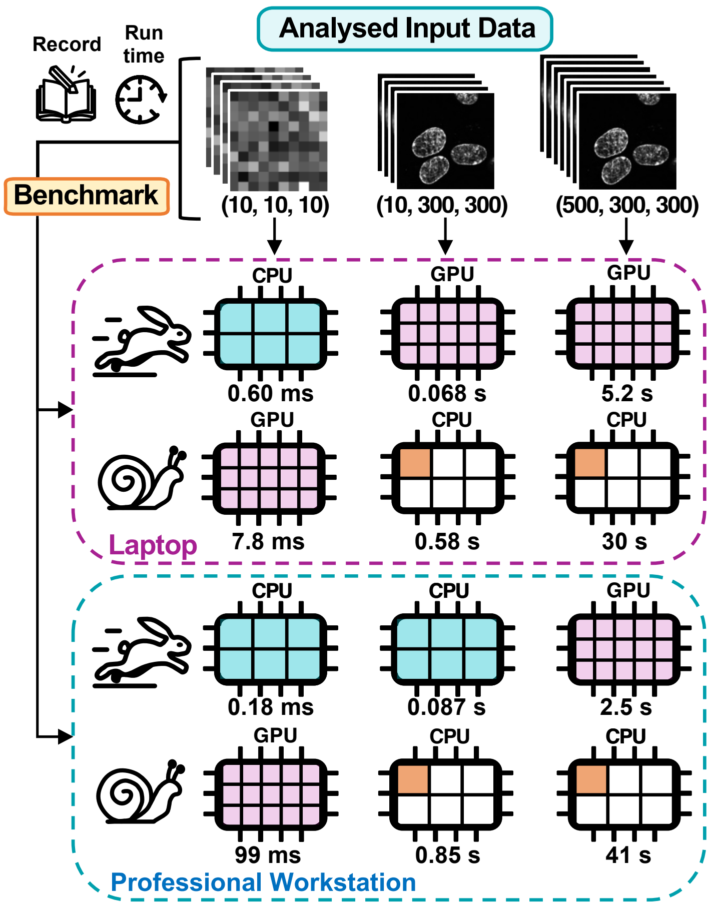


**Figure 2.2:** Comparative run times of multiple implementations of an algorithm, ran on either a consumer-grade laptop or a professional workstation. The fastest (rabbit) and slowest (snail) implementations depend on the dimensions of the input data and the user device. The underlying task carried out is a 5x frame-wise Catmull-Rom [89] upscaling. Different implementations are represented as processing chips with different colours. The implementation of both threaded (blue chip) and unthreaded (white chip with orange core)CPU uses optimised Cython code, while the implementation for GPUs (pink chip) is done through OpenCL.

As seen in Figure 2.2, for a small time-sequence image of 10 time-points with a size of 10 by 10 pixels, the fastest implementation of the example task on a consumer laptop is running in parallel on the CPU, with the slowest being using an OpenCL implementation running on the GPU. By increasing the image size to 300 by 300 pixels but keeping the same number of time-points, the fastest option in a consumer laptop is the OpenCL implementation, whilst the slowest is a single-thread implementation on the CPU. On the professional workstation, although the slowest is the same as in the laptop, the fastest implementation is still a parallelised CPU implementation. When increasing the number of timepoints to 500 and the size to 300 by 300 pixels, then the fastest in both devices becomes the GPU implementation.

Further quantitative studies were undertaken to comprehensively examine the performance characteristics of various implementations across diverse input datasets and parameter configurations on different hardware setups. The following Figure 2.3 depicts an obtained heatmap illustrating the runtime ratio for a 2D convolution performed using two distinct devices.

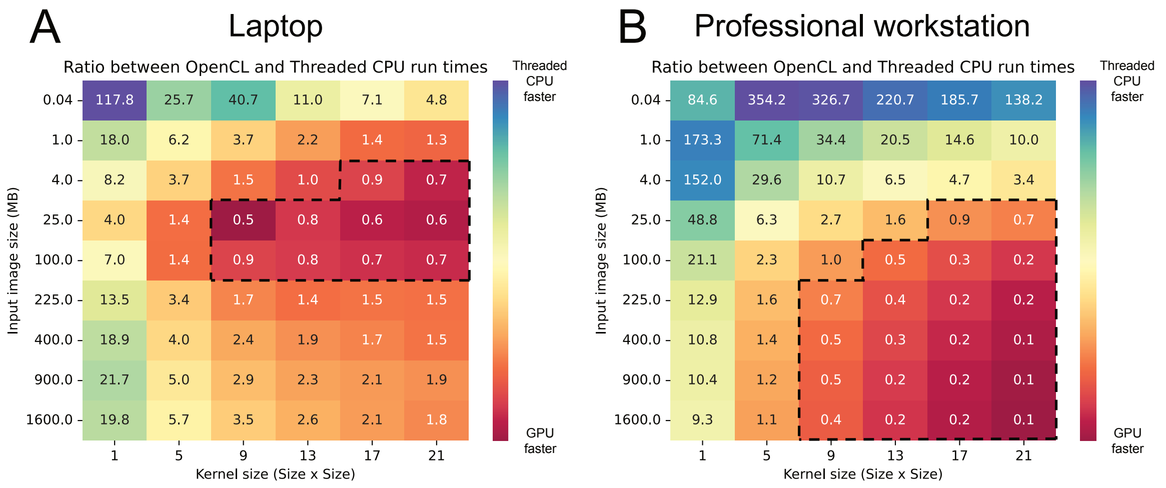


**Figure 2.3: Ratio between the run times of a 2D convolution.** Run times were measured across multiple input data sizes and kernel sizes using either a MacBook Air M1 (A) or a Professional Workstation (B). Areas within dashed lines correspond to kernel and image sizes where OpenCL is faster than threaded CPU.

The benchmark used features a 2D convolution with varying kernel sizes. The professional workstation results align with expectations - OpenCL implementation was markedly faster as input image size and kernel size increased. However, this was not mirrored on a laptop device. Laptop performance showed that while larger kernel sizes boosted OpenCL's relative efficiency against CPU threading, expanding image size beyond a certain threshold made the parallelised CPU approach faster again. Notably, this outcome likely ties to the test laptop (MacBook Air M1) lacking a dedicated GPU, demonstrating how closely run times are tied to specific user hardware. This apparent disparity in results underlines how reliance on one implementation can prove restrictive; for instance, choosing OpenCL implementation for lower-sized images could escalate the run time by up to 300 times compared to CPU processing. Similarly, threaded CPU processing for larger-sized images performed up to 10x slower than GPU processing on professional workstations.

Furthermore, an analogous investigation was replicated in Supplementary Figure 2.3, this time employing a Catmull-Rom [89] interpolator to upscale images by a factor of 5. The analysis encompassed the same laptop, the professional workstation, and also extended to Google Collaboratory to assess performance behaviour in a cloud-based environment. In contrast to the 2D convolution scenario, the Catmull-Rom interpolation task exhibited heightened computational demands. Consequently, the laptop's OpenCL implementation was faster compared to its CPU-based counterpart for data sizes that had previously favoured the CPU in the convolution task. Moreover, Google Collaboratory displayed a performance pattern similar to the laptop.

The following Figure 2.4 further elucidates the previous observations by plotting the run time of a

2D convolution task implemented in Cython unthreaded (single core CPU), Cython threaded (multi core CPU) with multiple schedulers, and PyOpenCL (GPU) across multiple image sizes, in the laptop and professional workstation.

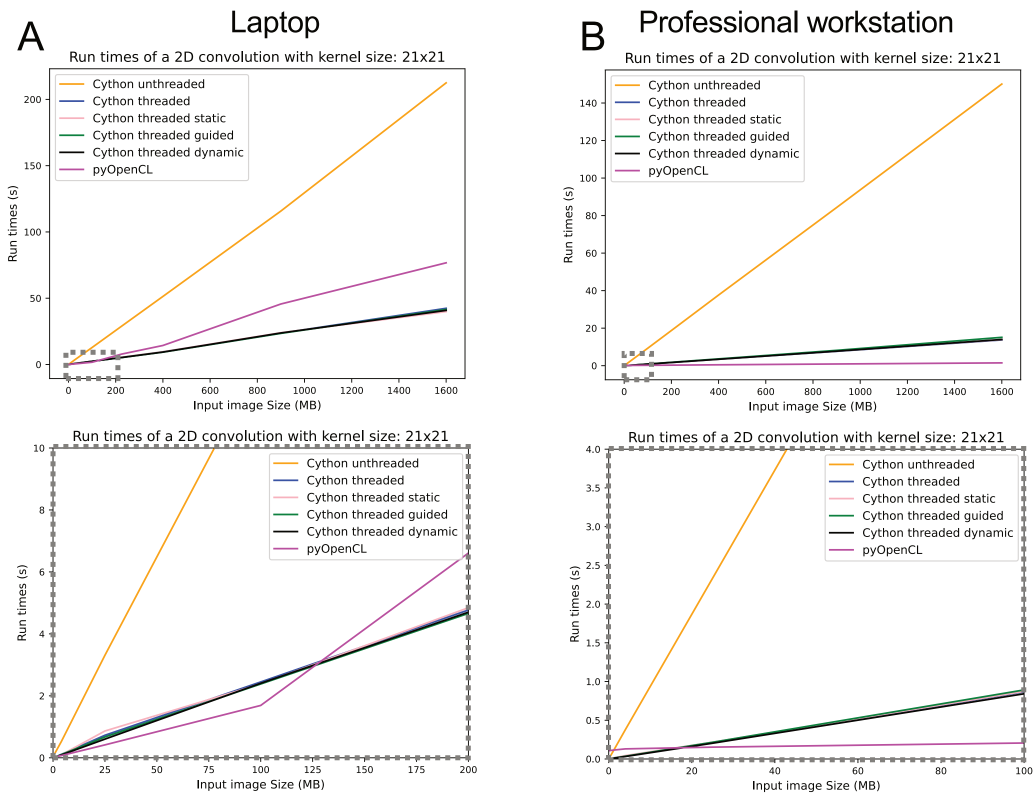


**Figure 2.4: Run time of each implementation is highly dependent on the shape of input data.** A 2D convolution was performed on images with increasing size using either a MacBook Air M1 (A) or a professional workstation (B). A 21 by 21 kernel was used in all operations.

Figure 2.4, particularly the zoomed-in regions, demonstrate the points in input image sizes from which the fastest implementation changes. For instance, in the laptop, the PyOpenCL implementation is the fastest until 125MB, after which the Cython threaded implementations become significantly faster. In the professional workstation, while unthreaded is virtually always the slowest implementation, the threaded implementations are only the fastest until the size increases to 20MB, after which PyOpenCL becomes the fastest. A similar study was conducted in Supplementary Figure A.3, where the kernel size for the convolution task was varied instead of the image size. The obtained results were similar to the previous analysis, in which unthreaded was virtually always the slowest implementation, and the threaded implementations were only the fastest until the kernel size increased to 20MB, after which

PyOpenCL becomes the fastest.

Collectively these findings stress the importance of having an adaptable system that selects the optimal implementation based on the image analysis task, input data and also the unique user hardware configurations.

To address these challenges, the Liquid Engine was introduced into NanoPyx. This machine learning-based system manages multiple tasks by exploiting various device components and selecting the most efficient implementation based on input data. The Liquid Engine features three main components: 1) meta-programming tools for multi-hardware implementation (named tag2tag and c2cl); 2) an automatic benchmarking system for different implementations; and 3) a supervisor machine learning-based agent that determines the ideal combination of implementations to maximise performance (see Figure 2.5).

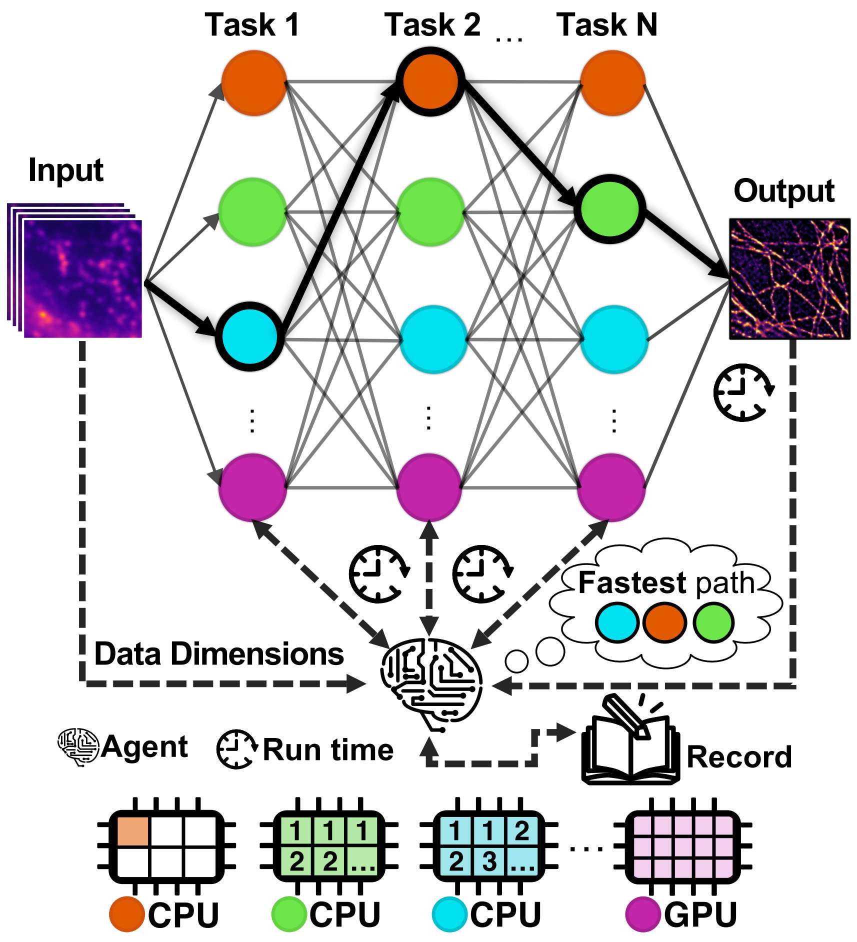


**Figure 2.5:** NanoPyx achieves optimal performance by exploiting the Liquid Engine self-optimisation capabilities. The image analysis workflows of NanoPyx are built on top of the Liquid Engine, which automatically benchmarks all implementations of all tasks in the specific workflow. The Liquid Engine keeps a historical record of the run times of each task and the shape of the used input, allowing a machine-learning-based agent to select the fastest combination of implementations. In the case of an unexpected delay, the agent dynamically adjusts the preferred implementations to ensure optimal performance.

### Meta-programming tools for multi-hardware implementations

Meta-programming is a programming technique where a program can manipulate or generate code during runtime [91]. In NanoPyx, meta-programming is used to generate multiple implementations of the same task, which can be run on the device's GPU orCPU. The Liquid Engine uses two meta-programming tools: tag2tag and c2cl (see Figure 2.6).

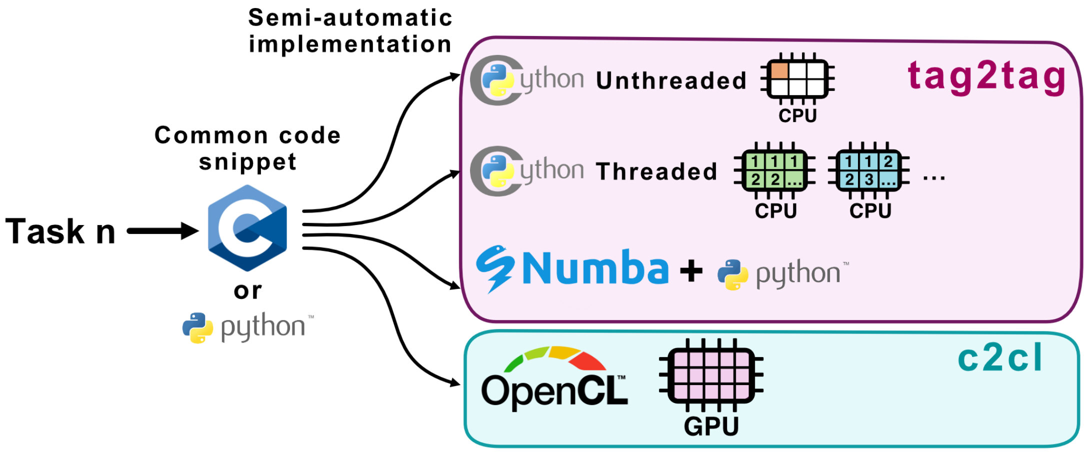


**Figure 2.6:** Meta-programming tools of the Liquid Engine. The tag2tag tool enables developers to generateCPU-based implementations. The c2cl tool generates GPU-based implementations using OpenCL.

The tag2tag tool enables developers to generate multiple implementations of the same algorithm written as C or Python code snippets. Effectively, tag2tag transcribes these snippets into single-threaded and multi-threaded versions of the code, generally then called by Cython.

Specifically, developers can write a single version of the code and delimit the "tag" to be propagated to the other implementations (e.g. a function). The tag2tag tool is able to read the content of the file, identify the tags, and store them in a dictionary, where the tag name is the key, and the associated code snippet is the value. After choosing the "tag", in the same script, the developer can create a `tag - copy`, where they specify what part of the original tag should be replaced, and what to replace it for. Tag2tag will identify the tag placeholders, and apply the specified replace commands to the associated tag code. It then replaces the tag placeholder in the file with the modified tag code. This tool is intended for use with Python (.py), Cython (.pyx), and OpenCL (.cl) files. In practice, for most tasks implemented in the Liquid Engine, a single-threaded version of the code was used as the original tag, and created multi-threaded versions of the code by replacing the `range` in the `for` loops with `prange`. Additionally, implementations with different schedulers for the parallelisation in the CPU were added.

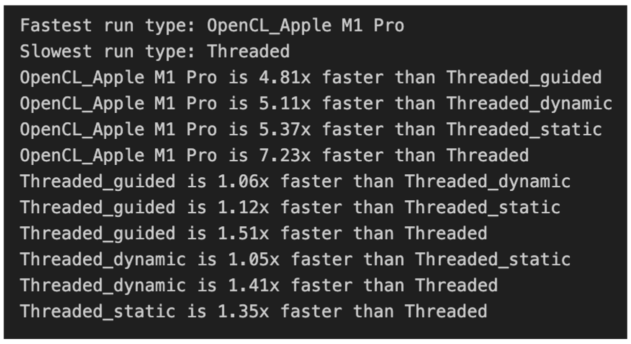
This allows the developer to easily create as many code variations as they find necessary. This approach streamlines the process of maintaining consistency across various code implementations, as altering the code in one version ensures that the modification is seamlessly and consistently applied to all other relevant versions. As a result, developers can effectively manage code updates and improvements, as these are propagated into the other implementations effortlessly, reducing redundancy and enhancing code maintainability.

Another meta-programming tool used in the Liquid Engine is the c2cl tool. c2cl is able to interpret the structure of functions, including the limits of the function (where it starts and where it ends) and its variables. It then is capable of writing a header for the functions and replacing the type of variables into a type that is compatible with the GPU. This allows the extraction of C functions and propagation into .cl files, so the C functions can be used in OpenCL kernels.

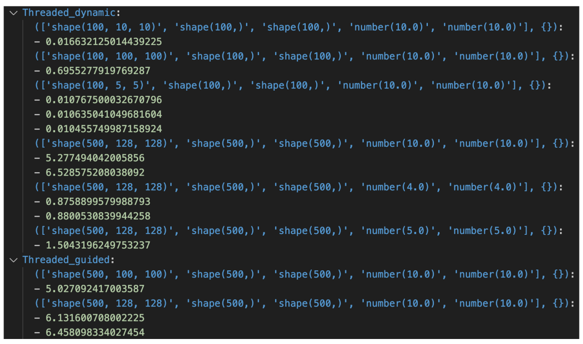
The Liquid Engine also supports Numba as an alternative performance-boosting option for Python code snippets. With all these implementations, NanoPyx can be run and used by users with diverse hardware configurations.

### Automatic benchmarking system

The Liquid Engine is capable of automatically benchmarking all implementations of the same task for the user's input. When benchmarking, the Liquid Engine runs the task using all available implementations and records the run times for each of them. It then automatically creates a hidden file in the user's device, where it stores both the run times and the dimensions of the used data. The automatic benchmarking system facilitates future comparisons and aids decision-making in selecting the optimal implementation. The following Figure 2.7 shows an example of the output of benchmarking a task (left), and an example of part of the information that a hidden file contains (right).



```
Fastest run type: OpenCL_Apple M1 Pro
Slowest run type: Threaded
OpenCL_Apple M1 Pro is 4.81x faster than Threaded_guided
OpenCL_Apple M1 Pro is 5.11x faster than Threaded_dynamic
OpenCL_Apple M1 Pro is 5.37x faster than Threaded_static
OpenCL_Apple M1 Pro is 7.23x faster than Threaded
Threaded_guided is 1.06x faster than Threaded_dynamic
Threaded_guided is 1.12x faster than Threaded_static
Threaded_guided is 1.51x faster than Threaded
Threaded_dynamic is 1.05x faster than Threaded_static
Threaded_dynamic is 1.41x faster than Threaded
Threaded_static is 1.35x faster than Threaded
```



```
Threaded_dynamic:
  (['shape(100, 10, 10)', 'shape(100,)', 'shape(100,)', 'number(10.0)', 'number(10.0)'], {}):
  - 0.016632125014439225
  (['shape(100, 100, 100)', 'shape(100,)', 'shape(100,)', 'number(10.0)', 'number(10.0)'], {}):
  - 0.6955277919769287
  (['shape(100, 5, 5)', 'shape(100,)', 'shape(100,)', 'number(10.0)', 'number(10.0)'], {}):
  - 0.010767500032670796
  - 0.010635041049681604
  - 0.010455749987158924
  (['shape(500, 128, 128)', 'shape(500,)', 'shape(500,)', 'number(10.0)', 'number(10.0)'], {}):
  - 5.277494042005856
  - 6.528575208038092
  (['shape(500, 128, 128)', 'shape(500,)', 'shape(500,)', 'number(4.0)', 'number(4.0)'], {}):
  - 0.8758899579988793
  - 0.8800530839944258
  (['shape(500, 128, 128)', 'shape(500,)', 'shape(500,)', 'number(5.0)', 'number(5.0)'], {}):
  - 1.5043196249753237
Threaded_guided:
  (['shape(500, 100, 100)', 'shape(500,)', 'shape(500,)', 'number(10.0)', 'number(10.0)'], {}):
  - 5.027092417003587
  (['shape(500, 128, 128)', 'shape(500,)', 'shape(500,)', 'number(10.0)', 'number(10.0)'], {}):
  - 6.131600708002225
  - 6.458098334027454
```

**Figure 2.7:** Example of benchmarks by the Liquid Engine. (Left) Output of benchmarking a specific task with a specific data input. (Right) Example of hidden file locally stored, containing the shape of the data used as input, and the run times of a specific run type (implementation).

#### Selection of fastest implementations

A typical bioimage analysis workflow is composed of multiple methods, such as drift correction, quality control, among others. A full workflow can then be integrated into the Liquid Engine, where individual tasks are ran sequentially. The selection of the fastest implementations for each task of a workflow is managed by a machine learning-based agent, where the agent predicts what are the optimal implementations of each task of the workflow.

```
WF = Workflow(task1(input), task2(output_task1), task3(output_task1, output_task2))
```

The Workflow class in the Liquid Engine is designed to facilitate seamless integration of new tasks within the framework. Each task within the workflow was configured to accept inputs derived from the outputs of previous tasks, the original input data, or a combination of both. This modular architecture allows for efficient data flow and ensures that each task can use the required information generated by preceding tasks.

When queried by the workflow, the agent picks the fastest implementation for each task, sequentially. Through automatic benchmarking of each implementation, the Liquid Engine keeps an historic record of runtimes for each implementation. Whenever a workflow is scheduled to be run, the supervisor agent is responsible to select the optimal implementation based on the previous recorded run times. The agent can adapt to unexpected delays in any implementation. A schematic representation on how the agent's delay management works is depicted in Supplementary Figure A.4. In case a severe delay is detected, reaching a level where it could potentially lead to a different implementation becoming faster, the agent predicts if the optimal implementation has changed. For that, the agent predicts the likelihood of the delay being repeated in the future and then assigns a probability for each implementation that depends on an estimation of the expected run time that each one might take.

For instance, if the fastest implementation for a method uses OpenCL and the GPU is under heavy load, resulting in an abnormally prolonged run time that is longer than the second fastest, the agent activates its delay management. All available implementations are now assigned a probability that is a function of their expected run time, as given by the average values measured in the past. The expected run time for the delayed implementation is adjusted based upon the probability that the delay is maintained and the magnitude of the measured delay itself (see the Methods section for the mathematical description). Therefore, in this example, the probability the agent chooses to run using OpenCL is low, especially if the delay is continuously maintained. However, the delayed implementation should always have a bigger than zero probability to be chosen. Due to this probabilistic approach, the agent will still select and use the delayed implementation from time to time. This ensures that it can detect when the delay is over. Once it detects that the delay is over, the agent goes back to selecting implementations based only on the fastest average run time. Figure 2.8 shows resulting data from the agent's delay management system, obtained with an artificial delay induced by overloading the GPU with superfluous calculations in a separate Python interpreter.

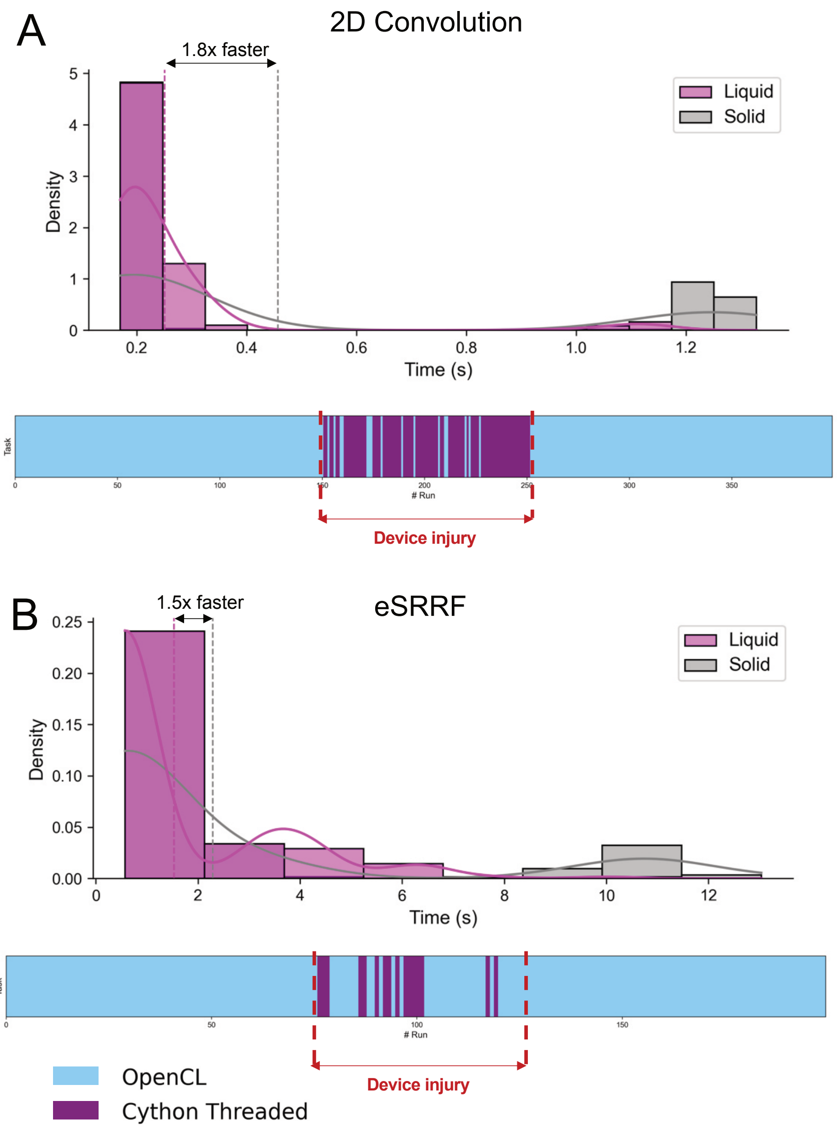


**Figure 2.8: Example of delay management by the Liquid engine.** Multiple 2D convolutions (A) and eSRRF analysis (B) were run sequentially in a professional workstation. Starting from two initial benchmarks, the agent is responsible to inform the Liquid Engine is what is the best probable implementation. Runs between red dashed lines represent the timeline where an artificial delay was induced.

Figure 2.8 shows that the Liquid Engine was able to detect an artificial delay that slowed down the OpenCL implementation. During the delay, the OpenCL implementation was used less times, but stochasticity allowed the Engine to detect the end of the delay. In this example, over the course of several sequential runs of the same method, it was shown that delay management improved the average run time by a factor of 1.8 for a 2D convolution and 1.5 for an eSRRF analysis.

Users can also manually initiate benchmarking, prompting the Liquid Engine to profile the execution of each implementation, using either multiple automatically generated data loads or using their own input, and identify the fastest one. This benchmarking is performed per task, allowing the Liquid Engine to adapt to the user's hardware configuration and progressively optimise the chosen combination of implementations to reduce the total run time. The system analyses similar benchmarked examples from the user's past data, using fuzzy logic [92] to identify the benchmarked example with the most similar input properties, utilising it as a baseline for the expected execution time. This system enables NanoPyx to immediately make adaptive decisions based on an initially limited set of benchmarked examples, progressively learning, and improving its performance as more data is processed.

### 2.3.2 The NanoPyx Framework

NanoPyx provides multiple bioimage analysis methods that can be combined into a comprehensive bioimage analysis framework for super-resolution microscopy. Figure 2.9 contains the results of applying NanoPyx implemented methods to the obtained dataset.

NanoPyx offers image registration methods, such as channel registration, which is used to align different channels of the same image, and drift correction, to correct any chromatic aberration and drift that might occur during image acquisition. These are both based on the Java implementations from NanoJ, and are currently implemented in Cython.

Then, NanoPyx implements fluctuation-based methods to generate super-resolution images: SRRF and eSRRF. Both of these methods are fully implemented into the Liquid Engine. In particular, the implementation of eSRRF is described in detail in the Annex A.2.2.

To ensure the fidelity of the reconstructions, rigorous quality control tools were introduced into NanoPyx. These include the Error Map of SQUIRREL to quantitatively assess errors introduced by the reconstruction algorithm, and FRC and Decorrelation Analysis (as in [73]) to determine image resolution. The implementations from the Error Map and the FRC are based on the NanoJ implementations. Currently, these methods are only implemented in Cython.

#### NanoPyx distribution

NanoPyx was developed with the primary objective of ensuring accessibility and ease of use for end users. To achieve this goal, three distinct interfaces were made available, through which users can interact with and utilise NanoPyx. Firstly, NanoPyx is accessible as a Python library, which can be conveniently accessed and installed via the GitHub repository for the latest development versions or through PyPI (Python Package Index) for stable releases. The Python library primarily targets developers seeking to incorporate NanoPyx's methodologies into their workflows. Secondly, "codeless" Jupyter notebooks were provided through the GitHub repository. These notebooks offer separate implementations of each method as individual notebooks. The term "codeless" denotes that users are not required to interact with any code directly: by sequentially executing cells, a graphical user interface (GUI) is

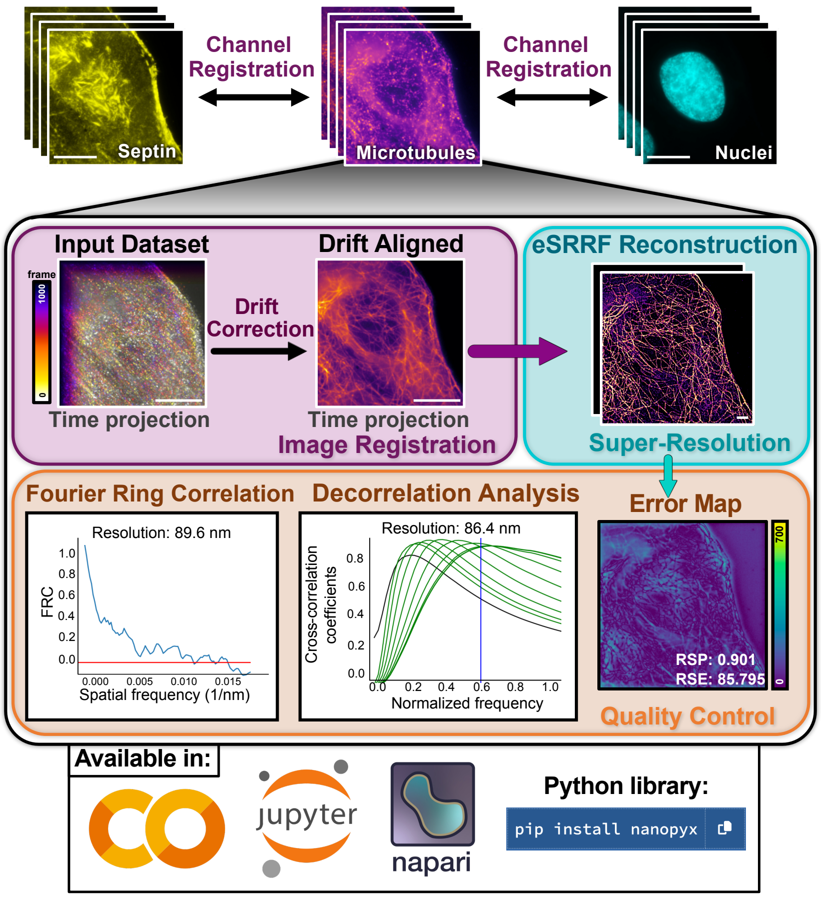


**Figure 2.9:** Microscopy image processing workflow using NanoPyx methods. NanoPyx implements several methods of super-resolution image generation and processing. Through NanoPyx, users can correct drift that occurred during image acquisition, generate a super-resolved image using enhanced radiality fluctuations (eSRRF) [67], assess the resolution of the generated image using Fourier Ring Correlation (FRC) [72] or Image Decorrelation Analysis [73], perform artifact detection using the error map and then perform channel registration in multi-channel images. NanoPyx methods are made available as a Python library, a napari [87] plugin, and Jupyter Notebooks that can be ran locally or through Google Colaboratory. Scale bars: 10 $\mu$m.

generated, enabling users to fine-tune the parameters for each step easily. Consequently, these notebooks are specifically designed for users with limited coding expertise. Lastly, for users desiring a more interactive approach, it is being developed a plugin for napari, a Python image viewer, granting access to all currently implemented NanoPyx methods. By offering these three diverse user interfaces, NanoPyx can be readily used by users irrespective of their coding proficiency level.

## 2.4 Discussion and Future Perspectives

NanoPyx introduces a novel approach to optimise performance for bioimage analysis through its machine learning-powered Liquid Engine. This enables dynamic switching between implementations to maximise speed based on data and hardware. By selecting the optimal implementation, NanoPyx achieves over 10 times faster processing (as observed in Figure 2.2). This has significant implications given the rapidly expanding scale of microscopy image datasets.

The Liquid Engine's optimisation strategy diverges from traditional approaches of relying on single algorithmic implementations. Alternative Python tools like Transonic [93] and Dask [94] that accelerate workflows through just-in-time compilation or parallelism do not adapt implementations based on context. In contrast, the Liquid Engine continually benchmarks and collects runtime metrics to train its decision-making model. It is this tight feedback loop that empowers dynamic optimisation in NanoPyx.

Beyond performance, NanoPyx provides an accessible yet extensible framework covering key analysis steps for super-resolution data. Workflows integrate essential functions like drift correction, reconstruction, and resolution assessment. NanoPyx builds upon proven ImageJ plugins while migrating implementations to Python. The modular architecture simplifies integrating components into new or existing pipelines. Currently, the only methods fully implemented into the Liquid Engine are the spatial analysis of SRRF and eSRRF.

The implementation of eSRRF as a Liquid Engine workflow has been comprehensively elucidated and evaluated in Appendix A.2.2. In this regard, the choice to integrate eSRRF as a single task within the Liquid Engine stems from the recognition of a substantial overhead when the GPU serves as the fastest execution platform. Specifically, splitting each simple step as an individual task within the workflow would introduce considerable overhead due to the necessary data transfer between the CPU and GPU. This highlights the necessity for a delicate balance between the dynamic switch between implementations and the associated overhead linked to CPU-GPU data exchange. Currently, based on initial assessments conducted on specific machines, the decision was taken to consolidate complex full methods (like SRRF and eSRRF) into single tasks. This choice aims to mitigate the significant overhead encountered. However, it's important to emphasise that this decision was influenced by a certain bias, as the tests were performed on computers with access to OpenCL, which might not be a universal condition across all laptops. Looking ahead, the plan is to integrate the remaining NanoPyx methods (drift correction, channel alignment, error mapping, FRC, and decorrelation analysis) as individual tasks within the Liquid Engine for the same aforementioned reasons. This approach offers the advantage of seamless integration of all of these tasks into a Liquid Engine workflow, thereby forming a complete super-resolution analysis pipeline. Users can simply select the methods to be executed on the input image, build them into a workflow, and execute them. The Liquid Engine will then dynamically determine the most efficient execution strategy for each method.

A noteworthy future strategy to further explore performance enhancement via the Liquid Engine involves a departure from the sequential execution of tasks within a workflow. Instead, an innovative approach could involve running an entire workflow in parallel, capitalising on the combined processing powers of both the computer's GPU and CPU. This approach envisions scenarios where distinct tasks operate concurrently, with some tasks using the GPU's capabilities while others leverage the CPU's resources. By embracing this strategy, the computational potential of the user's device can be maximally harnessed, potentially leading to substantial efficiency gains.

In summary, NanoPyx offers an innovative technique to enhance the efficiency of bioimage analysis while maintaining a modular and user-friendly design. This optimisation strategy with the Liquid Engine could have applications beyond microscopy, addressing other complex computational tasks. As data scales expand, NanoPyx offers researchers an actively improving platform to execute demanding microscopy workflows.

# 3

# Simulating SMLM: Evaluating Performance and Guiding Experimentation

**Contents**

This chapter provides a comprehensive explanation of the developed simulations implemented in NanoPyx. In addition, this chapter focuses on the use of the employed simulations to comprehensively explore some of the state-of-the-art localisation algorithms in SMLM. Finally, the chapter presents a conceptual analysis of the the innovative experimental approach for SMLM introduced in the following chapter.

The outcomes stemming from the simulations concerning the performance of the SMLM algorithms are meticulously analysed. The insights gained from the simulations and models are then leveraged in the subsequent chapter, which is dedicated to experimental work.

## Contributions

Inês Cunha, the thesis author, developed the particle simulations, their use for assessing the localisation algorithms performance, and the formulation of the immunolabelling process model. Furthermore, Inês Cunha conducted comprehensive discussions on the resultant findings.

## Code availability

All the developed simulations can be found in the NanoPyx github repository `https://github.com/HenriquesLab/NanoPyx`.

## 3.1 Introduction and Objectives

SMLM is a powerful technique that surpasses the diffraction limit and is widely employed in biological studies. In SMLM imaging experiments, fluorophores undergo transient stochastic activation, followed by individual imaging and localisation [14]. This is achieved by fitting a PSF to each emitter using algorithms such as WLS or the MLE [95, 96].

Simulations play a pivotal role within the field of SMLM, aiding researchers in the meticulous design of experimental frameworks [97]. Furthermore, simulations provide valuable perspectives on localisation algorithms, creating controlled settings with known ground truth positions. This allows for thorough assessments of algorithm performance, error identification, and even facilitates the comparison of various algorithms or software solutions [34, 98].

Over the years, several software have emerged to address the diverse simulation requisites of the SMLM community. These include packages or plugins such as SureSim [99], SMeagol [100], Teststorm [101], and FluoSim [102].

In essence, these simulators follow a consistent sequence of steps: 1) create a ground truth of the structure to be simulated; 2) randomly distribute particles across the structure, with a greater probability of alignment with the previously established ground truth; 3) convolve the particles with a PSF; 4) generate N frames with the activation of the photoswitching of the particles [97, 98].

In particular, the particle positioning step can be done in several ways. A possible approach to do this is by using a Monte Carlo method [103], which uses randomness in a system to evolve and approximate specific quantities, without the need to find an analytical solution for the system. In the context of SMLM simulations, this method is particularly useful for generating particles according to a Probability Density Function (PDF), equivalent to the ground truth. The algorithm can randomly distribute particles and assign higher probabilities to positions that closely match the expected PDF. It then can iteratively adjust particle positions based on the assigned probabilities and the introduced randomness. As the Monte Carlo simulation progresses, particle positions should gradually converge towards a configuration that closely approximates the desired ground truth arrangement. [104, 105].

The development of a particle simulator within the context of this project holds significant importance. As outlined in Section 1.1.4, the accuracy of SMLM fitting algorithms in counting molecules faces challenges when dealing with high local emitter densities [29, 62]. This phenomenon arises from the overlapping PSF of neighbouring emitters, causing the localisation algorithm to merge them and introduce "merging artifacts". Previous simulations have demonstrated that these algorithms can reliably recover molecule positions up to a density of around 1 molecule per squared micrometer [29]. To extend this limit, more robust algorithms like multi-emitter fitting have been introduced, theoretically capable of accurately counting molecules up to densities of 10 molecules per squared micrometer [106]. Simulations have been instrumental in evaluating these algorithms, enabling the quantification of the maximum molecular density that the localisation algorithm can handle before merging artifacts occur. Considering the forthcoming experimental characterisation of localisation algorithms in the next chapter, it becomes crucial to perform simulations under matching reconstruction parameters and conditions. This not only holds the potential to provide more nuanced quantitative insights but also offers assistance in selecting the most suitable algorithm for subsequent experimental data analysis. Additionally, integrating the simulator into the NanoPyx platform enhances its accessibility and usability within bioimage analysis workflows, benefiting the wider scientific community.

Moreover, the upcoming chapter 4 will introduce a novel experimental setup designed explicitly to control the density of emitters per image, thereby facilitating the experimental characterisation of the SMLM reconstruction algorithms. This methodology is rooted in the concept of competitive labelling, which holds the potential to offer a linear means of controlling labelling densities. It is important to establish a comprehensive conceptual understanding of whether this new approach yields effective control and how it compares to the conventional approach of serial dilution, involving variations in the concentration of labelled antibodies. Therefore, a dedicated section within this chapter will delve into a thorough exploration of competitive labelling, elucidating its principles and drawing comparisons with conventional labelling strategies.

Hence, the upcoming sections will cover the development of a particle simulator for SMLM within the NanoPyx platform. It will then be be applied to quantitatively estimate the accurate counting abilities of various localisation algorithms (MLE, WLS and with multi-emitter fitting). Additionally, a study on the immunolabelling process will be conducted, involving a comparative analysis between the conventional and competitive labelling methods in terms of their efficacy in controlling labelling densities within a sample. These insights will be comprehensively discussed and leveraged into the subsequent experimental chapter.

## 3.2 Methodology

### 3.2.1 Particle simulator

A particle simulator was developed within the NanoPyx framework using the Monte Carlo method for particle placement based on a user-defined PDF. The simulator accepts any 2D image as the input PDF representing the ground truth structure. It has adjustable parameters including maximum number of particles, minimum distance between particles, maximum number of tries, and mean distance threshold between particles. The devised algorithm randomly generates particle positions with a probability proportional to the intensity of the input PDF at that location. It iterates, adding more particles and adjusting their positions based on the PDF and distance criteria, until stopping conditions are met. These include reaching the max number of particles, mean distance between particles drops below a threshold, or max tries is exceeded.

To visualise the particles, a function renders gaussians at the particle positions. It computes the gaussian integral at each pixel using the error function (erf) and accumulates the values to generate the final image. The key steps are the following.

- Accept 2D image as input PDF
- Randomly generate particles based on PDF intensity
- Iteratively add and adjust particles based on PDF, distance criteria
- Stop when max particles, distance threshold, or max tries is reached
- Render gaussians at particle positions using erf

The algorithms were coded in C, Cython and Python using Microsoft Visual Studio Code. Testing was conducted with pytest and nox against Python 3.9, 3.10 and 3.11 on both OSX (ARM64) and Linux (AMD64 on Ubuntu LTS) to ensure cross-platform compatibility.

### 3.2.2 Localisation Algorithm Characterisation

To evaluate the particle counting accuracy of SMLM algorithms, the simulator was used in an automated workflow:

1. Generate random ground truth images with varying molecular densities
2. Simulate particles for each ground truth using the simulator
3. Run simulated images through ThunderSTORM in ImageJ using PyImageJ [107]
4. Extract number of detections by the algorithm
5. Plot mean detections over ground truth molecular density

The algorithms assessed were WLS and MLE fitting, with and without multi-emitter fitting. Parameters matched the experimental chapter:

- FOV: 20x20 pixels, 14x14 pixel central region
- Pixel size: 100 nm
- Gaussian PSF: sigma 1.5 pixels, amplitude 3
- WLS/MLE: integrated Gaussian PSF, radius 3 pixels, sigma 1.5 pixels
- Multi-emitter: max 3 molecules, p-value 1e-6

This automated workflow generated a quantitative characterisation of the algorithms' counting accuracy across different molecular densities. The results will be leveraged in the next chapter on SMLM experiments.

All simulations were conducted in a professional workstation with an Intel i9-13900K, a NVIDIA RTX 4090 with 24Gb of dedicated video memory, a 1TB SSD and 128Gb of DDR5 RAM.

## 3.3 Results

Several simulations were designed to support the development and optimisation of super-resolution imaging techniques into the NanoPyx platform. These include a particle simulator, responsible for emulating particle positions atop the ground-truth input; a photoswitching tracks simulator, responsible for simulating fluorophore blinking through a three-state transition model (on, off, and bleached), thus enabling the creation of an image stack that reproduces stochastic particle blinking; and a noise simulator, tasked with generating Gaussian and Poisson noise for the image stack. This section will delve into the particle simulator and photoswitching tracks simulator. Furthermore, the particle simulator will be applied in a comprehensive study on of well-established SMLM algorithms, allowing a quantitative assessment of their performance. This analysis will take into account parameters that mirror those used in the subsequent chapter, facilitating a direct and comparison between simulation outcomes and experimental results. Moreover, a subsection will be dedicated to the conceptual analysis of the competitive labelling approach, which will be employed in the subsequent experimental chapter.

### 3.3.1 Particle Simulations

A Monte Carlo-based algorithm was developed within the NanoPyx platform to simulate two-dimensional particle fields. This algorithm generates particle positions according to a user-defined input PDF while considering parameters like particle density and distance thresholds. The algorithm's iterative nature and stopping criteria ensure the creation of particle configurations that closely match the input PDF. A more detailed description of the implemented Monte Carlo-based algorithm can be found in Appendix A.3.

Upon applying the algorithm to the input PDF, a list of particle positions is produced, serving as a foundation for subsequent steps. Then, another function is introduced to translate these positions into a visual image. It uses the error function ($erf$) to calculate the Gaussian's influence for each pixel within each particle's vicinity, as the theoretical model elucidated in the introductory chapter (section 1.1.4). This rendering process is conducted along with several input parameters, including the desired Gaussian profile characteristics, such as the widths (sigma_x and sigma_y) and amplitude.

For clarity, the subsequent code snippet illustrates the process of calculating the Gaussian integral using the error function and the subsequent accumulation of these calculated values to generate the final rendered image.

```python
for j in range(y_start, y_end):
    Ey = 0.5 * (erf((j + 0.5 - yp) / sigma_y**2) - erf((j - 0.5 - yp) / sigma_y**2))
    for i in range(x_start, x_end):
        Ex = 0.5 * (erf((i + 0.5 - xp) / sigma_x**2) - erf((i - 0.5 - xp) / sigma_x**2))
        image[j,i] += amplitude * Ex * Ey
```

In addition to spatial positioning, a temporal photoswitching model was implemented to simulate fluorophore blinking kinetics in SMLM. A function was written to model a 3-state system of fluorophore states: ON, transient OFF, and permanent OFF (representing fluorophore bleaching). It accepts parameters including number of particles, how many timepoints (number of frames), and transition probabilities between states. Particles are initialised and then randomly transition between states over time based on the provided probabilities. An array of integer states is returned indicating each particle's state at each timepoint. This photoswitching tracks simulator can then be combined with the previous spatial positioning of particles. For this, an function was developed to render particle gaussians with tracks, which accepts the state array and renders a particle at a given timepoint if its state is ON. All other particles will be invisible for that frame.

This provides a comprehensive framework for generating simulated SMLM data encompassing both the spatial distribution and temporal dynamics of single molecules. By tuning parameters like transition probabilities, a wide range of fluorophore blinking behaviours can be replicated. The developed simulations can then be leveraged in several in silico investigations of SMLM systems.

Furthermore, the subsequent analysis covers a rigorous quantitative assessment of the effectiveness of super-resolution single-molecule localisation algorithms. This study will exclusively use the developed particle positioning simulator, disregarding the temporal dynamics of fluorophores. This is because the localisation algorithms' performance is primarily affected by the spatial distribution of particles, and not by their temporal dynamics. The outcomes of this analysis are elaborated in the following section.

### Localisation algorithms characterisation

The developed simulator was seamlessly integrated into an automated workflow, generating data on the detected molecule count by the localisation algorithms relative to the number of molecules in the ground truth. This framework was detailed in the Methodology section.

The following Figure 3.1 illustrates instances of randomly generated ground-truth structures, accompanied by their corresponding rendered simulated particle images and subsequent reconstruction achieved through fitting a PSF using the MLE algorithm.

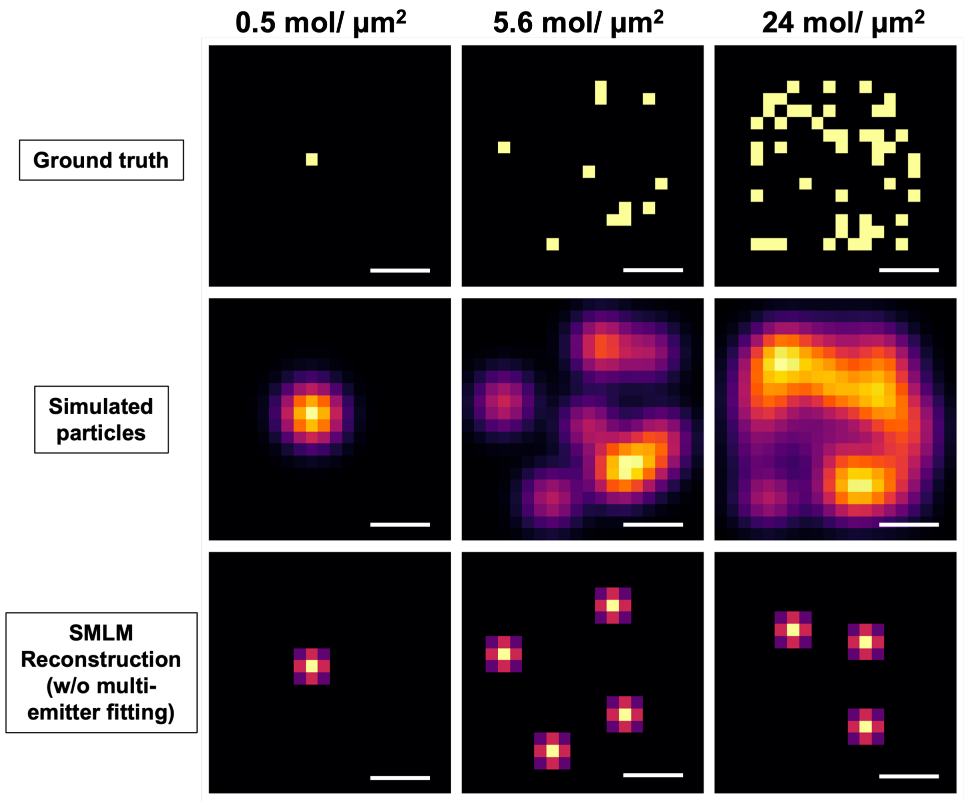


**Figure 3.1:** Example of particle simulation and consequent localisation. **(top)** Ground truth images at varying molecular densities (0.5 molecules per $\mu$m$^2$, 5.6 molecules per $\mu$m$^2$ and 24 molecules per $\mu$m$^2$); **(middle)** correspondent Simulated particle images, **(bottom)** Subsequent localisations achieved through fitting a PSF using the MLE algorithm. Scale bar: 500 nm.

Observing Figure 3.1, it becomes evident that the localisation algorithm accurately localised molecules within ground truths characterised by very low molecular densities (0.5 molecules per $\mu$m$^2$). However, as the molecular density increases, exemplified by a density of 5.6 molecules per $\mu$m$^2$, specific regions exhibiting a locally higher density pose challenges for the fitting algorithm in pinpointing individual emitters. This challenge often leads to the fusion of multiple molecules into one, causing merging artifacts. This is particularly pronounced in the ground truth scenario with 24 molecules per $\mu$m$^2$, where the localisation algorithm detects only a total of 3 molecules, a count even lower than the detection achieved in the case of the 5.6 molecules per $\mu$m$^2$ density.

A systematic approach involving the creation of ground truths, particle simulation, and subsequent reconstruction (outlined in the methods section) facilitated a comprehensive quantitative assessment of the algorithms. This assessment allowed for the depiction of detection counts achieved by various localisation algorithms in relation to the real number of molecules in the ground truth. The localisation algorithms studied were the WLS and MLE - with and without multi-emitter fitting. The localisation counts were normalised to represent values per squared micrometer. The Figures 3.2 and 3.3 present the localisation outcomes with the WLS and MLE, respectively.

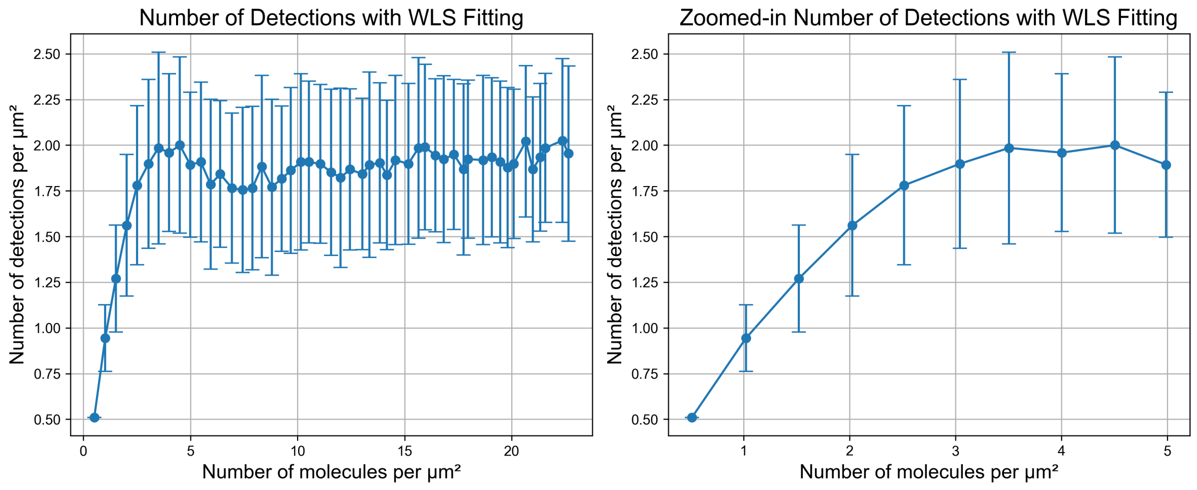


**Figure 3.2:** Number of detections with Weighted Least Squares. **(Left)** The average count of detections achieved through WLS fitting across 100 images is displayed, covering a range of 50 molecular densities (from 0.5 to 25 molecules per $\mu$m$^2$). The standard deviation corresponding to each density is also depicted. **(Right)** Zoomed-in view on densities between 0.5 and 5.6 molecules per $\mu$m$^2$ (a range of 10 densities) from the broader graph on the left.

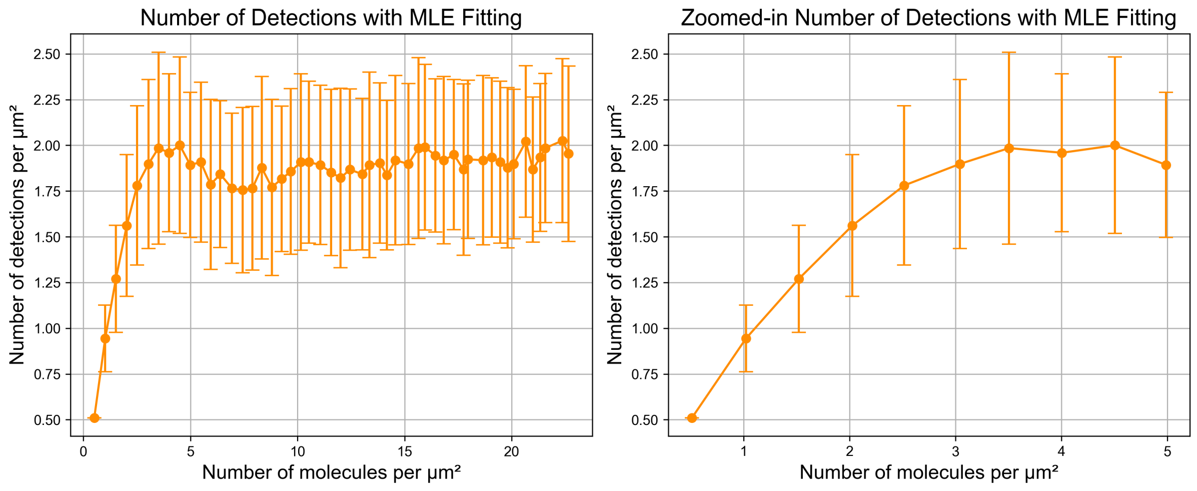


**Figure 3.3:** Number of detections with Maximum Likelihood Estimator. **(Left)** The average count of detections achieved through WLS fitting across 100 images is displayed, covering a range of 50 molecular densities (from 0.5 to 25 molecules per $\mu$m$^2$). The standard deviation corresponding to each density is also depicted. **(Right)** Zoomed-in view on densities between 0.5 and 5.6 molecules per $\mu$m$^2$ (a range of 10 densities) from the broader graph on the left.

From the observed patterns in the mean number of detections obtained through the fitting process for both algorithms (depicted in the left graphs of both Figures 3.2 and 3.3), a noticeable trend emerges. Initially, at very low molecular densities, there appears to be a proportional relationship between the number of detections and the number of molecules present in the ground truth. However, this linear relationship reaches its peak, undergoes a slight decrease, and subsequently levels off, resulting in a plateau.

The observed elevated standard deviations stem from the inherent randomness of the generated ground truths, representing molecules across the field of view (FOV). This randomness introduces the possibility of certain regions exhibiting higher local densities compared to others, or molecules being distributed uniformly throughout the FOV. Consequently, variations in merging occurrences emerge, leading to fluctuations in the number of detections. Remarkably, the algorithms reached a peak detection capacity of approximately 2.5 molecules per $\mu$m$^2$.

For a deeper insight into the graphs behaviour, a closer examination is offered through magnified views in the graphs on the right. These reveal additional subtleties, highlighting that the slope of the graph tends towards 1 for the lowest molecule count. As the real molecular count incrementally increases, the slope diminishes gradually, accompanied by a simultaneous increase in the standard deviation.

Specifically, up to a molecular density of approximately 1 molecule per $\mu$m$^2$, the number of detections showcases a linear correlation with the molecular count, and a comparatively lower standard deviation. Conversely, as molecular density surpasses this threshold, the slope of the number of detections starts descending, indicating a progressive decline in reconstruction accuracy.

Furthermore, the same study was conducted with the MLE algorithm, with multi-emitter fitting enabled. The results are depicted in Figure 3.4.

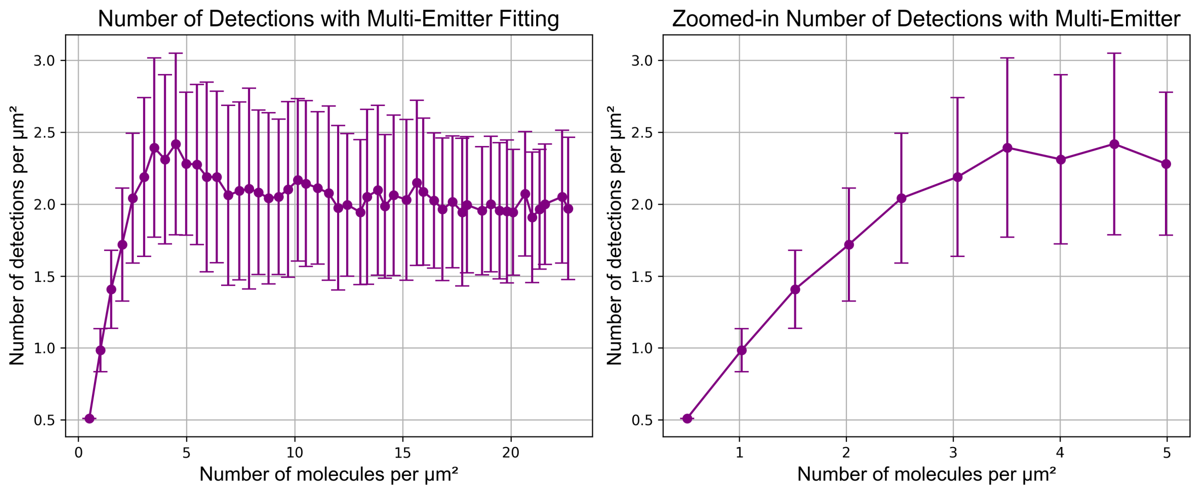


**Figure 3.4:** Number of detections with Multi-emitter fitting. **(Left)** The average count of detections achieved through MLE fitting with multi-emitter fitting across 100 images is displayed, covering a range of 50 molecular densities (from 0.5 to 25 molecules per $\mu$m$^2$). The standard deviation corresponding to each density is also depicted. **(Right)** Zoomed-in view on densities between 0.5 and 5.6 molecules per $\mu$m$^2$ (a range of 10 densities) from the broader graph on the left.

The results obtained from Figure 3.4 exhibit a comparable trend to the reconstruction outcomes achieved without employing multi-emitter fitting. The plot demonstrates a linear increase for very low labelling densities, followed by a peak that transitions into a plateau. Notably, the multi-emitter fitting algorithm showcases an enhanced detection capacity, reaching a maximum detection count at 3 molecules per $\mu$m$^2$. In the zoomed-in section of the plot, a proportional correlation between the number of molecules and detections is apparent (slope approximate to 1) up to 1.5 molecules per $\mu$m$^2$. This underscores the capability of multi-emitter fitting to accurately detect molecules within a broader density range, surpassing the performance of the non-multi-emitter fitting approach.

To better compare the behaviour between WLS, MLE and multi-emitter fitting, the mean detections of all three algorithms were plotted in the following Figure 3.5.

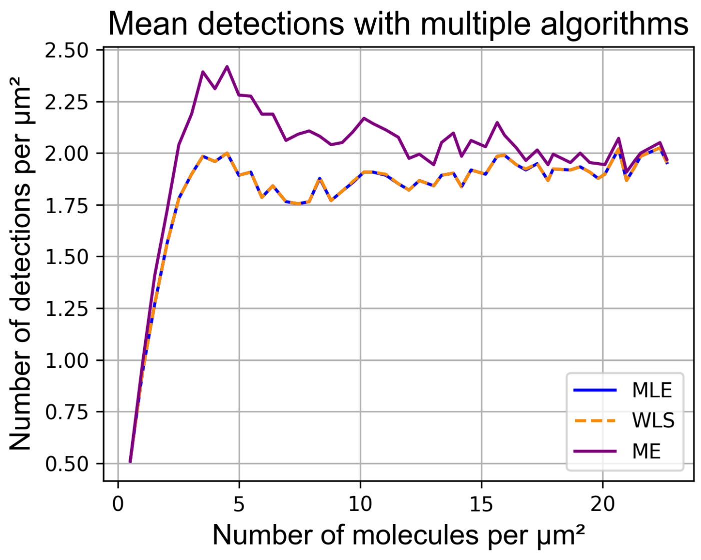


**Figure 3.5:** Mean detections achieved with WLS, MLE and multi-emitter (ME) fitting.

Figure 3.5 offers a clearer comparison of the tendencies of the studied algorithms. Notably, the WLS and MLE algorithms demonstrate visually identical tendencies. The cumulative error stemming from a comparison between the WLS and MLE algorithms was determined by quantifying the total number of differing detections between the two and dividing it by the overall number of detections. This calculation yielded an 8% error, highlighting the closely aligned performance of these algorithms. Between WLS and MLE, the latter displays a higher number of detections. Furthermore, when these algorithms are contrasted with the multi-emitter approach, it becomes evident that enabling multi-emitter fitting permits a superior detection capacity. However, while the multi-emitter approach achieves a higher peak detection count, this count eventually converges with the counts of the other algorithms as the molecule count increases.

Furthermore, an assessment of computational performance was conducted alongside this investigation. The WLS algorithm outperformed the MLE algorithm by 1.5 times in terms of speed. Notably, the introduction of the multi-emitter fitting approach resulted in a 3-fold increase in computation time compared to the MLE algorithm.

### 3.3.2 Conceptual Competitive Labelling

In section 1.1.4 of the general introduction, it was introduced the concept of antibody-antigen binding particularly for the case of indirect immunolabelling. As previously stated, the epitope - antibody binding is related to the antibody concentration according to the Hill-Langmuir equation, where the fraction of bound antibodies is given by the following equation 3.1.

$$P(labelled) = \frac{[Ab]}{[Ab] + K_d} \tag{3.1}$$

Whereas $[Ab]$ is the antibody concentration and $K_d$ is the dissociation constant.

A central goal of this project is to experimentally characterise the precision of SMLM algorithms in counting molecules. This requires a linear control over the molecular density in the sample, enabling the quantification of algorithmic detections across specific density levels. Such analysis aids in determining the algorithm's upper limit for accurate counting. However, achieving precise control over emitter density is an intricate task. This complexity arises from the non-linear relationship between the binding probability of epitopes to labelled antibodies and the antibody concentration. Consequently, traditional serial dilution of antibody concentration does not yield a linear variation in the number of labelled molecules within the structure, as governed by the previous equation 3.1 (see Figure 3.6).

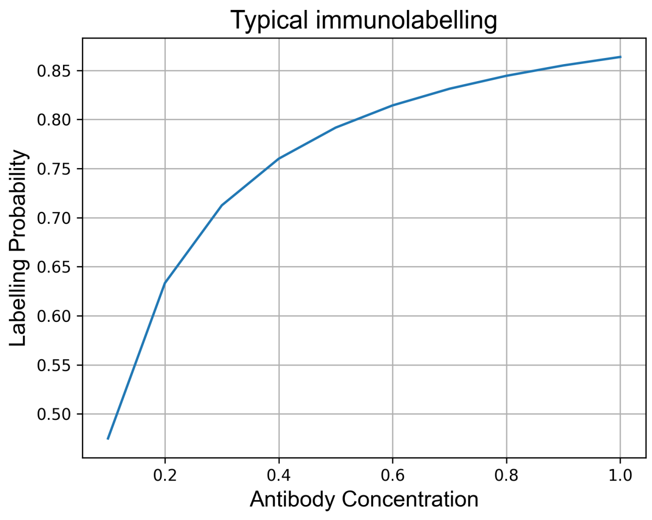


**Figure 3.6:** Labelling probability with typical immunolabelling. With serial dillution, the probability of the epitope being bound to the labelled antibody does not vary linearly with the antibody concentration. It depends on the dissociation constant $K_d$. For this graph, it was used a $K_d$ of 0.1.

To overcome this, an alternative immunolabelling approach can be employed to attain a linear control over the molecular density in a sample. This novel strategy, entitled competitive labelling, involves using an unlabelled antibody as a competitor alongside the labelled antibody, as depicted in Figure 3.7.

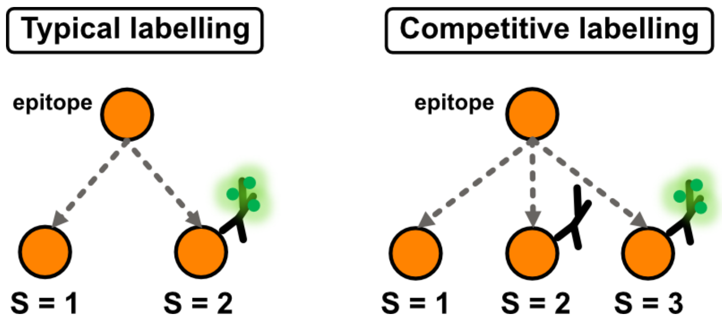


**Figure 3.7:** Visual comparison between typical and competitive immunolabelling. **(Left)** In typical immunolabelling, the epitope of the protein of interest is targeted by the antibodies conjugated with the fluorophores (labels), yielding 2 possible states: S = 1 (unbound epitope) and S = 2 (bound epitope). **(Right)** In competitive labelling, there are three possible states: S = 1 (unbound epitope), S = 2 (bound epitope with unlabelled antibody) and S = 3 (bound epitope with labelled antibody).

Observing Figure 3.7, in theory, when the total concentration of antibodies (both labelled and unlabelled) remains constant and sufficiently high to saturate the entire structure, the state in which no antibodies are bound to the epitope (S = 1) becomes negligible. The unlabelled and labelled antibodies naturally compete for binding to the protein of interest's epitope. Consequently, the modelled system should encompass two viable states: S = 2, representing epitope binding with the unlabelled antibody, and S = 3, signifying epitope binding with the labelled antibody. Assuming equal binding probabilities for both antibodies, the probability of epitope binding with the labelled antibody will be the following.

$$P(labelled) = \frac{[Ab_{labelled}]}{[Ab_{unlabelled}]} \tag{3.2}$$

Whereas $[Ab_{labelled}]$ and $[Ab_{unlabelled}]$ are the concentrations of labelled and unlabelled antibodies, respectively. Effectively, this implies that manipulating the ratio between the concentrations of labelled and unlabelled antibodies will inherently lead to a proportional variation in the probability of the epitope binding to the labelled antibody. This correlation is visually illustrated in Figure 3.8.

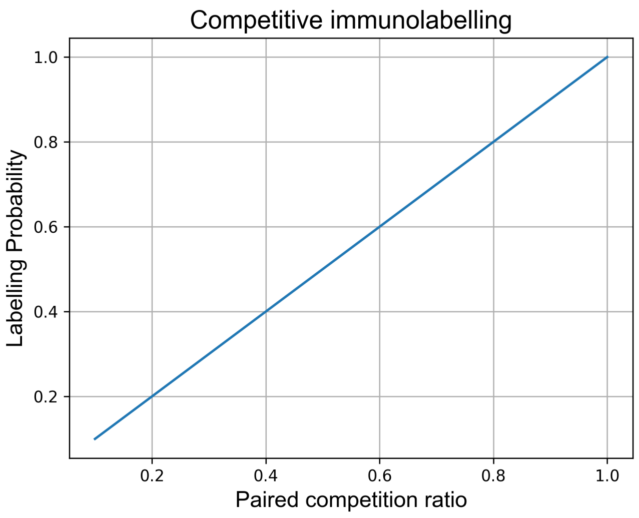


**Figure 3.8:** Labelling probability with competitive immunolabelling. With competitive labelling, the probability of an epitope being bound by the labelled antibody varies linearly with the ratio between the concentrations of labelled and unlabelled antibodies.

Ultimately, a novel approach for controlling the label density within a sample has been conceptualised, involving the manipulation of the labelled/unlabelled antibody ratio.

## 3.4 Discussion and Future Perspectives

The particle simulations developed within the NanoPyx framework use the Monte Carlo method to generate particles, given a user-defined PDF. Notably, these simulations were designed to be easily accessible to the broader scientific community through the NanoPyx platform, allowing their integration into various bioimage analysis workflows for various applications. A significant innovation lies in the simulator's flexibility, as it does not impose any constraints on the choice of ground truth—unlike most existing simulators that either impose predefined ground truths or require users to create specific ones within the simulator's framework. This distinctive feature empowers users to input any image of their choosing, thereby expanding the potential applications of these simulations. For instance, users could employ real experimental PDF data as input and subsequently simulate multiple experimental scenarios based on that input image. This offers the potential for a more accurate and realistic representation of experimental conditions. Consequently, researchers can gain enhanced insights into specific parameters they intend to fine-tune for their experiments.

Subsequently, the developed simulator was employed to conduct a quantitative analysis of algorithms in the context of SMLM. While it is not unprecedented to use simulations for this purpose, the distinct contribution of this study lies in the valuable quantitative insights it offers. These insights are anticipated to play a pivotal role in the forthcoming experimental chapter, where the same reconstruction parameters will be employed.

The results yielded from the analysis of the MLE and WLS (Figures 3.3 and 3.2) fitting algorithms revealed results in line with the expectations. The algorithms exhibit accurate performance at sparse molecular densities, corresponding to a linear regime; yet, as density increases, the algorithms begin to merge molecules, leading to the observed saturation effect. Both algorithms show very similar performances, as evidenced by the cumulative error of 8% between the two. This similarity can be attributed to their shared objective of optimising the fitting of the same designated PSF to the data, despite their distinct mathematical underpinnings. Furthermore, the linear operational range within which these algorithms can reliably count molecules aligns with a molecular density of approximately 1 molecule per $\mu$m$^2$. This is consistent with the established theoretical limit.

The analysis was extended to encompass MLE fitting in combination with the multi-emitter fitting option, tailored to enhance reconstructions for elevated molecular densities. The results exhibited the expected trend: the algorithm's labelling density threshold for accurate reconstruction surpassed that of the non-multi-emitter variant. This demonstrates the outperformance of the multi-emitter fitting approach in terms of detection capacity. Nevertheless, even with this enhancement, the algorithm demonstrated a linear regime (slope of approximately 1) for accurate detection up to 1.5 molecules per $\mu$m$^2$. This goes against the theoretical value of 10 molecules per $\mu$m$^2$, possibly due to the parameters chosen for the reconstruction. Furthermore, as the molecular density increases, its detection count converged with that of the non-multi-emitter algorithms, as depicted in Figure 3.5.

This analysis distinctly reveals the labelling density points at which the reconstruction algorithms begin to exhibit limitations, thereby enriching the quantitative comprehension of their operational boundaries. Notably, it's crucial to highlight that the quantifiable insights extracted from these simulations are dependent on various factors, such as the chosen reconstruction parameters, the simulated PSF defined by integrated Gaussians (including its dimensions), among others. Importantly, these parameters were tailored to match the conditions expected for application in the forthcoming experimental chapter, as opposed to being optimised to determine conditions that would yield optimal algorithm performance.

Furthermore, an evaluation of the computational time required for the execution of these algorithms within the simulation framework was conducted. This study holds particular relevance due to its applicability in the subsequent analysis of experimental data. Notably, the scale of experimental data is considerably larger than that of the simulated data, thereby enhancing the computational demands in the context of dataset size and algorithmic complexity. This escalation in computational demands reinforces the significance of computational efficiency in the context of bioimage analysis workflows, highlighted in the previous chapter. The results demonstrated that the WLS algorithm exhibited the fastest computational performance compared to the MLE counterpart. Enabling multi-emitter fitting significantly increased the computational time (by a factor of 3), as expected due to the inherent complexity of the algorithm. Although the multi-emitter fitting approach exhibited the best performance, its computational requirements render it less practical for integration into the experimental chapter. Given the closely matched quantitative results between WLS and MLE fitting, the decision was made to employ the WLS algorithm without multi-emitter fitting in the experimental phase, primarily due to its superior computational efficiency.

Moreover, a simple conceptual model was developed to illustrate the concept of competitive labelling. Using competitive labelling is anticipated to enable precise control over labelling densities within a sample by manipulating the molar ratio between labelled and unlabelled antibodies. This newfound control promises to facilitate analogous analyses of localisation algorithms within experimental setups, a task that has posed challenges in the past. It is important to highlight that this study operates on two underlying assumptions: 1) the total concentration of the antibodies is sufficiently high to saturate the entire structure, and 2) the binding probabilities of the labelled and unlabelled antibodies are equal. These assumptions will be taken into account during the analysis of results in the next chapter.

4

# Competitive Labelling Experiments

## Contents

This chapter introduces a novel methodology for biochemically controlling labelling densities in SMLM experiments. It covers sample preparation, image acquisition, and data analysis methods. The obtained results, both qualitative and quantitative, are discussed in detail, offering insights into the methodology's efficacy. The implications of these findings are explored, along with potential future directions for research in this field.

**Contributions:**

The author of the thesis, Inês Cunha, conducted all the experiments and subsequent analyses described in the following sections.

## 4.1 Introduction and Objectives

It has been shown that localisation algorithms face challenges in accurately determining the positions of emitters when their local density becomes too high [29,62]. In particular, as demonstrated by simulations in the preceding chapter, both the commonly employed MLE and WLS algorithms exhibited a threshold for the maximum density of concurrently active fluorophores in an image frame, which was found to be around 1 emitter per square micrometer. This aligns with established findings in the literature [95].

Evaluating the performance of these algorithms is a relatively straightforward task within simulated scenarios, as the real molecule count is known in the ground-truth [108]. However, characterising the behaviour of these algorithms within real experimental setups represents a substantial obstacle, primarily due to the inability of linearly controlling the density of emitters within the sample [14,48]. This makes it inherently challenging to accurately quantify the number of molecules within a given structure.

Quantifying the practical density threshold at which these algorithms effectively operate within real experimental scenarios could provide valuable insights for optimal imaging conditions and ensuring the accurate quantification of molecular counts [109].

Previous studies have successfully manipulated the labelling densities within samples to assess the resulting quality of super-resolved images. For example, Gibbs et al. (2015) [110] employed a method wherein they mixed conjugated and unconjugated antibodies (with and without fluorescent labels) at varying molar ratios, thus enabling precise control over labelling density. However, their study primarily focused on comparative analyses of the final reconstructions, without providing any quantitative insights. Moreover, Gibbs et al. (2016) [111] adjusted emitter density by tuning the imaging buffer for several photoswitching dyes, but did not quantify the number of detected molecules. Furthermore, Cox et al. (2017) [48] developed an analytical tool for identifying reconstruction artifacts stemming from high emitter density. However, their work didn't directly determine the reliable density limit at which SMLM algorithms perform in real experiments. They manipulated labelling densities in the sample through illumination, a common approach to control emitter numbers [61], showcasing qualitative artifact formation at high densities. Nevertheless, this method lacks linear control, as laser intensities don't exhibit proportional relationships with photoswitching rates of commonly used fluorophores [51]. In addition, Manley et al. (2018) [112] introduced an autonomous illumination control system capable of evaluating excessive density of photoswitching molecules and subsequently adjusting the illumination accordingly. Yet, they do not directly characterise the limitations of the SMLM algorithm. Alternative methods involve adjusting the number of acquired time frames ( [109]) to assess algorithm detections. However, this relies on fluorophore photobleaching rates rather than achieving varying densities within a single image frame.

Although these studies show promising strategies of controlling the number of activated molecules in experimental settings, these do not directly characterise the algorithmic limits within which precise molecule counting can be achieved. This leads to the objectives of this chapter.

This chapter presents an innovative approach designed to achieve precise control over labelling densities in SMLM experiments, with the primary goal of comprehensively characterising localisation algorithms using real data. The proposed method employs competitive labelling, a concept described in the previous chapter (section 3.3.2). With this novel methodology, a mixture of labelled and unlabelled molecules are used to compete for available binding sites within the sample. The number of labelled molecules in the sample is then dictated by the ratio of labelled and unlabelled antibodies (depicted in Figure 4.1). This technique is applied to fixed cells, leveraging the well-defined microtubule structure as the cellular framework. The labelled antibodies are tagged with the widely employed fluorophore Alexa Fluor 647, while the unlabelled antibodies act as competitors. Subsequently, a thorough assessment of the localisation algorithm's performance is carried out across a range of emitter density conditions using the experimental dataset. This assessment encompasses various metrics, including the quantification of localisations, evaluation of error mapping for artifacts, and measurement of the super-resolved image resolution. The resulting data is subjected to comprehensive analysis, leading to a detailed exploration of the implications derived from these observations. Furthermore, this discussion explores potential avenues for future research in this specific field.

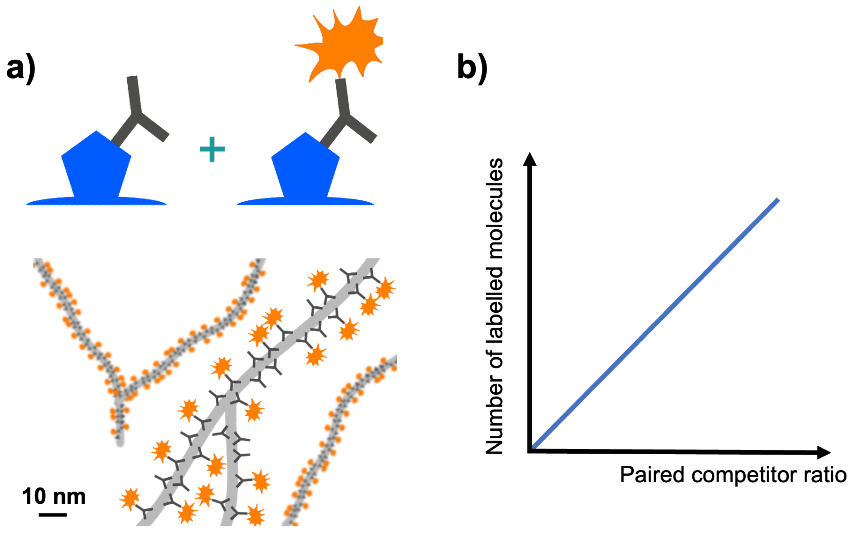


**Figure 4.1:** Competitive Labelling System Overview. a) Schematic of the competitive labelling concept, where a mixture of labelled and unlabelled antibodies competes to bind to tubulin. Notably, the illustration portrays direct immunolabelling, while the actual procedure employed indirect immunolabelling. b) Graphical depiction of the trend of labelled molecule count in a sample under varying ratios of labelled/unlabelled antibodies.

## 4.2 Methodology

This section outlines the methodology used to acquire the competitive labelling data and explains the subsequent data analysis approach.

### 4.2.1 Sample Preparation

The objective of these experiments was to perform immunolabelling of microtubules in fixed COS-7 cells.

#### COS-7 Cell Culture and Fixation Protocol

COS-7 cells were cultured in phenol red-free high-glucose, L-Glutamine containing Dulbecco's modified Eagle's medium (DMEM; Thermo Fisher Scientific) supplemented with 10% (v/v) fetal bovine serum (FBS; Sigma), 1% (v/v) penicillin/streptomycin (Thermo Fisher Scientific) at 37 ℃ in a 5% CO2 incubator. Cells were seeded on clean $\mu$-slide 8 well glass bottom: 1.5H (170 $\mu$m +/- 5 $\mu$m) D 263 M Schott glass (IBIDI) at a density of 30 000 cells per well. A volume of 300 $\mu$L was occupied in each well. Cells were grown for 24 hours before fixation.

Prior to the immunolabelling process, several buffers were prepared and stored appropriately. The buffers used during the immunolabelling procedure are described in the following tables.

**Table 4.1:** CB Buffer recipe

| Volume | Reagents | MW [g/mol] | [Stock] | [Final] |
|---|---|---|---|---|
| 90 ml | ddH2O | | 5x | 1.25x |
| 195.2 mg | MES | 195.24 | | 10 mM |
| 876.6 mg | NaCl | 58.44 | | 150 mM |
| 190.2 mg | EGTA | 380.35 | | 5 mM |
| 90.1 mg | Glucose | 180.2 | | 5 mM |
| 101.7 mg | MgCl2 | 203.3 | | 5 mM |
| 100 ml | Final Volume | | | |

After preparing the CB buffer with the specified reagents, the pH was adjusted to 6.1 using NaOH pellets, and the solution was filtered through Millipore "Stericup" 0.22 $\mu$m. 10 ml aliquots of the CB buffer were stored in 15 ml Falcon tubes at -20℃.

**Table 4.2:** Washing buffer recipe

| Volume | Reagents | MW [g/mol] | [Stock] | [Final] |
|---|---|---|---|---|
| 9.9 ml | PBS | | 1x | 1x |
| 100 ul | Tween 20 | | 10 % | 0.1 % |
| 10 ml | Final Volume | | | |

The washing buffer was then stored at 4ºC.

**Table 4.3:** Cell Blocking buffer recipe

| Volume | Reagents | MW [g/mol] | [Stock] | [Final] |
|---|---|---|---|---|
| 0.5 g | Bovine Serum Albinum | | | 5 % |
| 10 ml | PBS | | 1x | 1x |
| 10 ml | Final Volume | | | |

The cell blocking buffer was then stored at 4ºC.

Immediately before starting the cell fixation process, two CB buffer-derived solutions were prepared. The first solution, CB buffer 1, was prepared by adding 120 $\mu$l of glutaraldehyde (25% (v/v)) and 250 $\mu$l of Triton X-100 (10% (v/v)) to 9630 $\mu$l of CB buffer. The second solution, CB buffer 2, was prepared by adding 400 $\mu$l of glutaraldehyde (25% (v/v)) to 4600 $\mu$l of CB buffer. Both solutions were heated to 37ºC in a waterbath before use.

After completing the incubation period, the cells were carefully removed from the incubator and the media was delicately removed. Cells were then incubated for 2 minutes in CB buffer 1, followed by a 10-minute incubation with CB buffer 2. During the incubation period, a fresh 0.5% $NaBH_4$ solution was prepared by adding 25 mg of $NaBH_4$ to 5 ml of PBS. After the incubation period, the cells were quenched with 0.5% $NaBH_4$ for 7 minutes, and the solution was refreshed thrice during quenching. The cells were then washed three times with 1x PBS for 1 minute, 5 minutes and 10 minutes, respectively.

## Antibody Staining

The cells were blocked with the previously prepared Cell Blocking buffer for 30 minutes. They were then incubated with mouse anti-$\beta$-tubulin primary antibodies in blocking buffer (5 $\mu$l of 2 mg/ml stock of primary antibody to 1 mL of blocking buffer) for 1 hour at room temperature. After the primary antibody incubation, the cells were washed twice with 10% Tween-20 for 5 minutes each and then twice with 1x PBS for 1 minute each.

The next step was to incubate cells with the secondary antibodies. A distinct combination of secondary antibodies was used for every set of three wells of the chambers. Specifically, the unlabelled F(ab')2-Goat anti-Mouse IgG (H+L) Cross-Adsorbed Secondary Antibody (1.2 mg/mL stock) and F(ab')2-Goat anti-Mouse IgG (H+L) Cross-Adsorbed labelled with Alexa Fluor 647 (2 mg/mL stock) were employed during this incubation process, with varying molar ratios between them.

The volume required for the labelled and unlabelled antibodies was calculated based on the labelled/unlabelled ratio using the following equation:

$$\text{ratio} \times\ 5\ \mu\text{L (labelled)} + \text{(1 - ratio)} \times \frac{\text{2 mg/mL}}{\text{1.2 mg/mL}} \times x\ \mu\text{L (unlabelled)} + \text{Blocking buffer} = \text{1 mL}$$

A total of 10 different ratios were tested, ranging from 10% to 100% with increments of 10%. The following Table 4.4 shows the concentrations of labelled secondary antibodies (with AF647), unlabelled antibodies and blocking buffer used in each competitive labelling experiment.

**Table 4.4:** Concentrations of labelled secondary antibodies (with AF647), unlabelled antibodies and blocking buffer used in each competitive labelling experiment.

| Competition Ratio | Labelled AB ($\mu$L) | Unlabelled AB ($\mu$L) | Blocking Buffer ($\mu$L) |
|---|---|---|---|
| 10 % | 0.5 | 7.47 | 992.03 |
| 20 % | 1 | 6.64 | 992.36 |
| 30 % | 1.5 | 5.81 | 992.69 |
| 40 % | 2 | 4.98 | 993.02 |
| 50 % | 2.5 | 4.15 | 993.35 |
| 60 % | 3 | 3.32 | 993.68 |
| 70 % | 3.5 | 2.49 | 994.01 |
| 80 % | 4 | 1.66 | 994.34 |
| 90 % | 4.5 | 0.83 | 994.67 |
| 100 % | 2.5 | - | 497.5 |

Following the secondary antibody incubation, the cells were washed with 0.1% Tween-20 for 1 minute and then twice for 5 minutes each. Finally, they were washed twice with 1x PBS for 5 minutes each.

### Post fixation

For post fixation, the cells were fixed for 10 minutes using 4% paraformaldehyde in PBS. After fixation, they were washed three times for 10 minutes each using the previously prepared washing buffer.

The 8 well-chambers were then stored at 4ºC in 1x PBS until imaging.

## 4.2.2 Image Acquisition

For each chamber, cells were mounted with a GLOX-MEA buffer (50 mM Tris, 10 mM NaCl, pH 8.0, supplemented with 50 mM MEA, 10% [w/v] glucose, 0.5 mg/ml glucose oxidase, and 40 $\mu$g/ml catalase). Data acquisition was performed with the Nanoimager microscope (Oxford Nanoimaging; ONI) equipped with a 100 x oil-immersion objective (Olympus 100x NA 1.45). The FOV has a size of Fluorescence illumination was performed using the 640 nm laser in TIRF illumination at an angle of 53.7º to minimise background fluorescence.

With 3 % of the total laser power ON, adequate fields-of-view containing cells with a lower density of microtubules were selected for imaging. First, a TIRF image was taken, where 5 frames were acquired with an exposure time of 50 ms. Then, 100% of the laser power was used to promote blinking of the fluorophores in the sample. This corresponds to an approximate irradiance of 7.5 kW/cm$^2$ at the sample plane. The exposure time was set to 10 ms, and a total of 20 000 frames were acquired for each field of view. The pixel size of the images is 117 nm.

For each labelled/ unlabelled ratio, five cells (five fields-of-view) were imaged. Therefore, a total of 5 (cells) x 10 (ratios) = 50 image stacks were acquired and analysed.

### 4.2.3 Data Analysis

Following the data acquisition, each image stack was reconstructed into a super-resolved image and post-processed using the Fiji plugin ThunderSTORM. Due to the extensive nature of the acquired datasets, consisting of 20 000 frames for a single image, the manual processing times were prolonged. In fact, performing the PSF fitting and post-processing took approximately 1 hour per image stack. To streamline the process, automation was implemented by creating a macro in Fiji. The macro was built to process an entire folder of image stacks, performing all the specified steps and saving each output in another folder, automatically. This enabled the generation of super-resolved reconstructions efficiently. The macro consisted of the following steps:

1. Open the image stack from the specified folder.

2. Crop the image stack to a 200 x 200 pixels region of interest centered within the FOV. This not only reduces processing time but also ensures a more consistent excitation across the entire FOV, given the Gaussian illumination profile of the laser used.

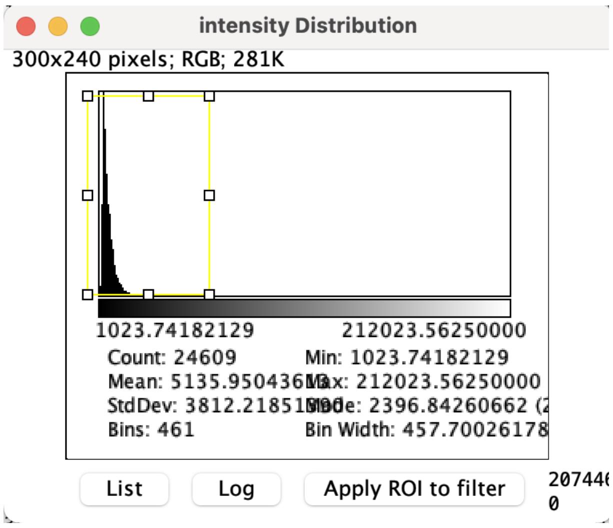
3. Run the analysis of the ThunderSTORM plugin for the 20000x200x200 image to reconstruct the stack into a super-resolved image. The camera setup was adjusted to a pixel size of 117 nm, and a base level of 200 counts to remove the background noise. Image filtering was performed with a wavelet filter (B-Spline) with an order of 3 and a scale of 2.0. To detect approximate molecule positions, the local maximum method was used with a peak intensity threshold of three times the standard deviation of the wave (3*std(Wave.F1)), and an 8-neighbourhood connectivity. For the subpixel localisation of the emitters, it was employed a PSF fitting with an integrated gaussian, with a fitting radius of 3 pixels and an initial sigma of 1.5 pixels. The used fitting method was the WLS, as it is one of the fastest methods available and offers a similar accuracy to the MLE [96]. Multi-emitter fitting was not enabled, due to its slow processing time. The results were presented as Average Shifted Histograms, and a 5 x magnification was chosen for the resulting images.

4. Apply drift correction to the super-resolved reconstruction, using the cross-correlation method.

5. Save the localisations table in a .csv file in the specified folder, along with the resulting super-resolved image.

Given the significant variation in the number of detections for each image stack in this study, the subsequent post-processing steps required manual intervention. This approach was necessary to ensure that no biases or errors were introduced during the process.

For each image, histograms of intensity, uncertainty, and sigma distribution of localizations were plotted. These histograms were thoroughly analysed, allowing for the identification of localisations with unrealistic parameters. Specifically, localizations with an intensity below 1000 or above 30000 (as depicted in Figure 4.2) were removed, along with those having an uncertainty exceeding 2 nm. Additionally, the sigma distribution plot (in nm) of the localizations was examined to identify any potential grid artifacts. Grid artifacts may manifest as spikes or peaks in the sigma plot, indicating systematic errors introduced during the imaging process.


```
intensity Distribution
300x240 pixels; RGB; 281K

1023.74182129                  212023.56250000
 Count: 24609            Min: 1023.74182129
 Mean: 5135.950436Max: 212023.56250000
 StdDev: 3812.21851Mode: 2396.84260662 (2
 Bins: 461               Bin Width: 457.70026178
```

**Figure 4.2:** Intensity distribution example in ThunderSTORM. The yellow region of interest represents the region of interest in the intensity histogram applied to the data.

It is important to emphasise that the filtering steps applied during the post-processing of the data were solely intended to remove unrealistic values and artifacts, ensuring the reliability of the final results. Careful consideration was given to striking the right balance in filtering to prevent over-removal of data points, which could potentially impact the results and introduce unintended biases. As such, options like density filtering were not used to avoid any distortion in the number of detections, which is the primary variable of interest in this study.

After completing the post-processing of all images, the study aimed to compare the number of detections between the processed images, i.e, with different labelling densities (varying labelled/ unlabelled ratios). Due to the inherent variability in the number of microtubules present in each cell, directly comparing the total number of detections between images would not provide meaningful results. To overcome this, for each super-resolved image with a specific ratio, the number of detections was analysed along a precisely measured 1 micrometer segment of a microtubule. To achieve this, a Region of Interest (ROI) was manually drawn (with the segmented line tool in ImageJ) along a microtubule, with a width of 8 subpixels to ensure accurate coverage and minimal background inclusion (see Figure 4.3).

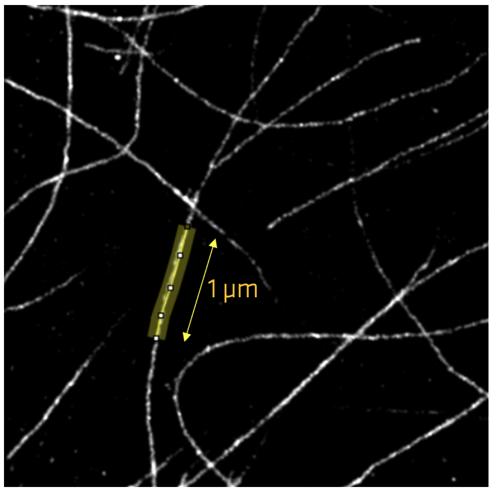


**Figure 4.3:** Selection of a specific area (with 1 $\mu$m in length) as the ROI in a super-resolved image of microtubules. By using the ThunderSTORM plugin, it is possible to assess the number of detections for this ROI.

The length of the hand-drawn segment was assessed, and the number of detections in this segment was then divided by the measured length to obtain the exact number of detections per micrometer.

To ensure a more robust analysis, the SNR of the number of detections was considered instead of just the raw detections count. This accounted for potential noise and detections resulting from nonspecific binding within the selected ROI in the microtubule. A squared micrometer area of noise (without any microtubules) close to the selected microtubule was considered. The number of detections in this noise area was used to divide the number of detections obtained in the microtubule area.

For each labelling density, a total of 3 cells was considered, and for each cell, 3 different microtubules were analysed, along with 3 different areas of noise. Thus, for each competitor ratio, a total of 3 x 3 = 9 measurements of detections (SNR) per micrometer were taken.

All computational analysis were conducted in a Macbook M1 Pro, with 16Gb of RAM and a 512Gb SSD.

## 4.3 Results

After conducting the immunolabelling experiments with varying competition ratios (ranging from 0.1 to 1.0 with increments of 0.1), the acquired data underwent post-processing and analysis, as detailed in the previous section. For analysis, a total of 3 different cells were considered for each ratio, meaning a total of 30 fields-of-view were analysed.

The following Figures show three of the acquired super-resolved images obtained for three specific ratios: 0.1 (10% labelled secondary antibodies and 90% unlabelled antibodies) (Figure 4.4), 0.5 (50% labelled antibodies and 50% unlabelled antibodies) (Figure 4.5), and 1.0 (100% labelled antibodies) (Figure 4.6).

It is important to note that the brightness and contrast of the images shown were adjusted to enhance the visualisation of the microtubule structures. These adjustments varied according to the intensity of each image, as reconstructions with higher percentages of labelled molecules exhibited a higher intensity compared to those with more unlabelled antibodies. However, all analytical procedures were carried out with unaltered data.

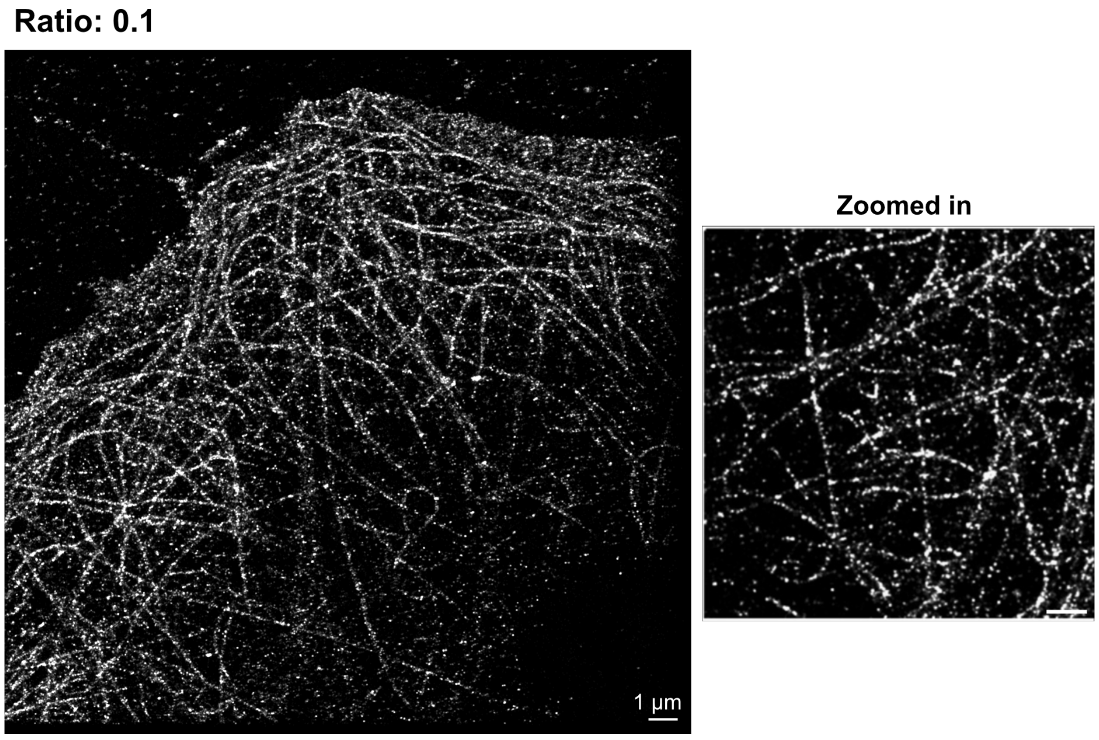


**Figure 4.4:** Obtained super-resolved reconstruction with a competition ratio of 0.1 (10% labelled secondary antibodies with Alexa Fluor 647, and 90% unlabelled antibodies). On the right, a zoomed-in region of the image is shown. Scale bar: 1 $\mu$m.

The super-resolved images obtained with a paired competitor ratio of 0.1 (Figure 4.4) demonstrated that the microtubule structure was not fully sampled, as anticipated, considering that only 10% of the secondary antibodies were labelled with Alexa Fluor 647. However, qualitatively, there is an absence in merging artifacts (i.e., "blurred areas" where neighbouring molecules were merged together) in the image, possibly indicating a higher accuracy of the algorithm used for localising individual emitters. Using a labelling ratio of 0.1 forces a lower density of emitters in each frame, which might have implied that the PSF of individual emitters did not significantly overlap. As a consequence, the localisation algorithm could accurately fit an integrated Gaussian to each individual emitter, without causing any visible merging artifacts.

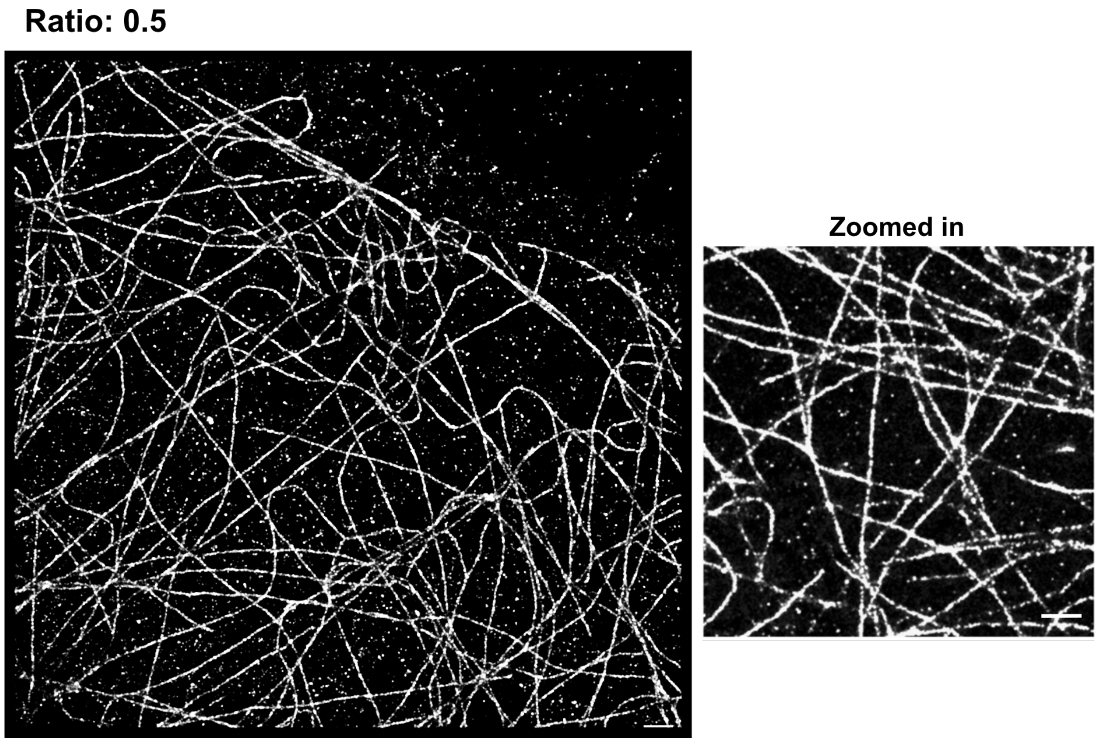


**Figure 4.5:** Obtained super-resolved reconstruction with a competition ratio of 0.5 (50% labelled secondary antibodies with Alexa Fluor 647, and 50% unlabelled antibodies).On the right, a zoomed-in region of the image is shown. Scale bar: 1 $\mu$m.

The super-resolved images obtained with a paired competitor ratio of 0.5 (Figure 4.5) showcased a clear visualisation of the microtubule structure. However, it was observed that in certain areas, the microtubules were still undersampled, indicating that the density of labelled molecules was not sufficient to fully capture the intricacies of the entire microtubule network. Despite some undersampling, the absence of visible merging artifacts in these images provides further evidence of the high fidelity of the algorithm used for localising individual molecules.

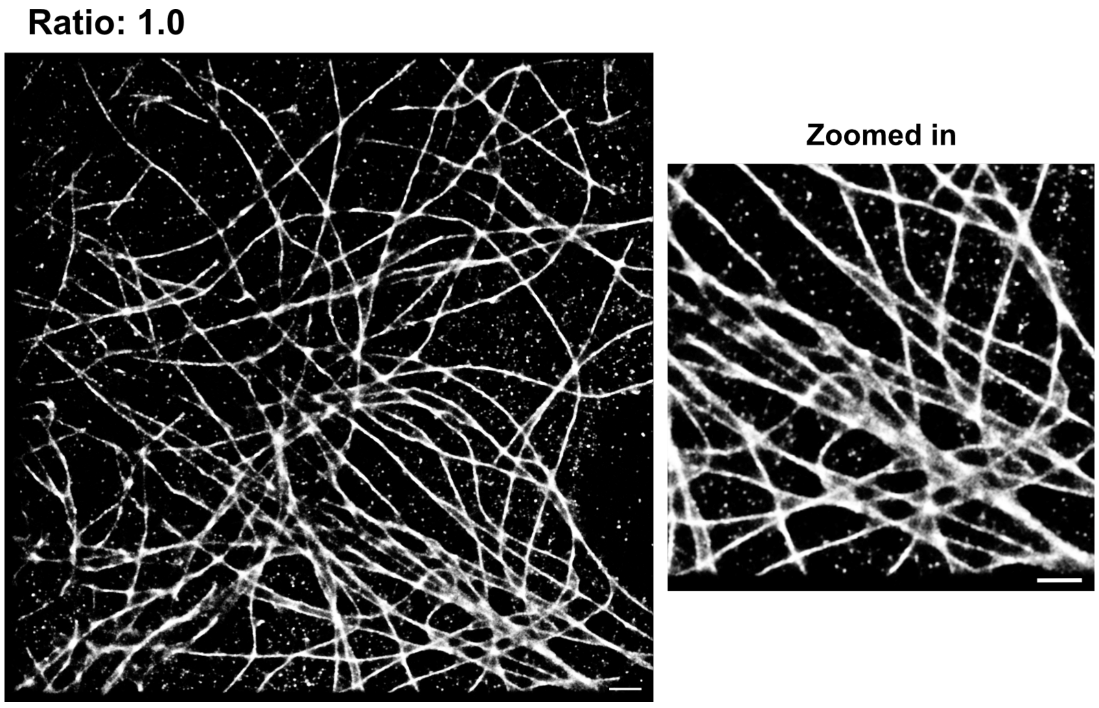


**Figure 4.6:** Obtained super-resolved reconstruction with a competition ratio of 1.0 (100% labelled secondary antibodies with Alexa Fluor 647).On the right, a zoomed-in region of the image is shown. Scale bar: 1 $\mu$m.

The results from Figure 4.6, depicting the super-resolved image obtained with 100% labelled antibodies, show no qualitative signs of undersampling in the microtubule structure. This indicates that the labelling density was sufficient to adequately capture the details of the microtubule network. However, a notable observation is the presence of merging artifacts in the image, particularly evident in the zoomed-in region. This suggests that the labelling density in each frame was too high for the employed algorithm to accurately and individually localise the molecules. As a consequence, some of the fluorescent emitters merged together, leading to the observed merging artifacts.

These results qualitatively show that there is a labelling density trade-off between adequately sampling the structure and avoiding merging artifacts.

In the subsequent analysis, the primary objective was to quantitatively determine the labelling density breaking point of the algorithms. In practice, this corresponds to finding the paired competitor ratio from which the number of detections per frame does not increase with the number of labelled molecules. For this, as explained in the previous section, the SNR of the number of detections along 1 micrometer of a microtubule was considered, for 3 different cells for each ratio.

A comprehensive explanation of the efficient calculation of Signal-to-Noise Ratio (SNR) for each ratio, along with all acquired detections in both microtubules and noise regions, is available in Appendix A.4.

With the obtained data from the Tables provided in the Appendix (A.1 and A.2), it is possible to plot the mean SNR for each paired competition ratio, along with the standard deviation (Figure 4.7).

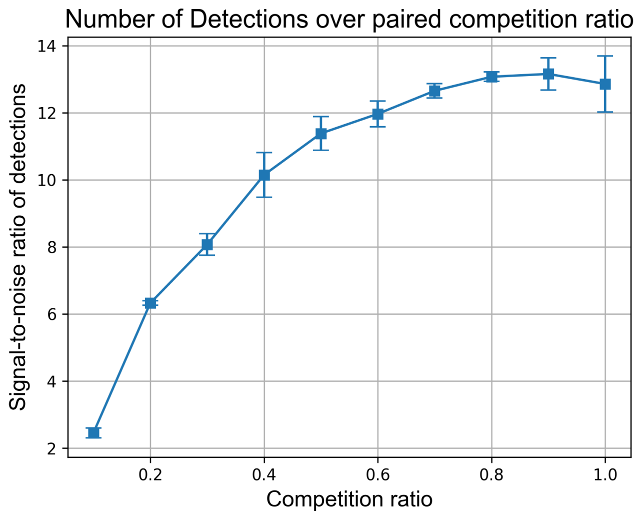


**Figure 4.7:** Mean SNR for each paired competition ratio. The error bars represent the standard deviation.

The obtained Figure 4.7 reveals insights into the relationship between labelling densities and the number of detected molecules in the conducted experiments. As the competition ratio increases, indicating a higher proportion of labelled molecules relative to unlabelled ones, it is observed a rise in the number of detected molecules (SNR) until a certain point. Beyond this critical threshold, the number of detections reaches a plateau and starts to decrease, which was expected. Moreover, the plot presents a slightly nuanced behaviour. For very low labelling densities (ratios of 0.1 to 0.2), it is observed a linear increase in the number of detections, approximately with a slope of 1. As the number of labelled molecules increases further, the slope of the plot diminishes, indicating a reduced rate of detection growth. This observation suggests that the algorithm might already be introducing merging artifacts for ratios of 0.3, 0.4, and so on, as the slope consistently decreases for higher ratios. Eventually, the slope approaches zero, reaching a plateau at a ratio of approximately 0.8. The sharp linearity between the SNR and the ratio is only evident at densities of 0.1 and 0.2, which lead to undersampling of the microtubule structure. This highlights a trade-off between the molecular undercounting by the algorithm and the undersampling of the biological architecture.

Precisely determining the optimal trade-off point from the graph alone becomes inherently complex, due the continuous decrease in the slope of the plot. Without further metrics, the results could be interpreted in two ways: either the ideal density point is at a lower ratio of 0.2, which coincides with undersampling of the microtubule structure but with algorithmic linearity; or at a ratio of 0.8, where the slope levels off, corresponding to the point from which algorithms are no longer counting additional molecules, and with no visual undersampling of the structure.

To address this subjective aspect and gain deeper insights, it was adopted an additional approach by assessing other metrics directly within the obtained super-resolved images. The objective was to obtain quantitative measures that could aid in identifying potential artifacts or undersampling more objectively. For this, the Error Map method available in NanoPyx was used, which provides two essential metrics: the RSE and the RSP. These metrics could potentially offer valuable quantitative insights into the performance of the super-resolution algorithm at different labelling densities. Additionally, the decorrelation analysis was employed, also implemented in NanoPyx, to assess the resolution of the super-resolved images. The error and Pearson's correlation metrics from the error map should reveal empirical knowledge into artifact creation, whereas the decorrelation analysis could potentially provide information about whether the resolution of the image is affected by the labelling density.

The obtained super-resolved reconstructions of the same 3 cells for each ratio were used in the study. For the error mapping of the images, it was used the obtained TIRF image for each reconstruction as the reference image. An example error map and the corresponding blurred super-resolved image are shown in Figure 4.8.

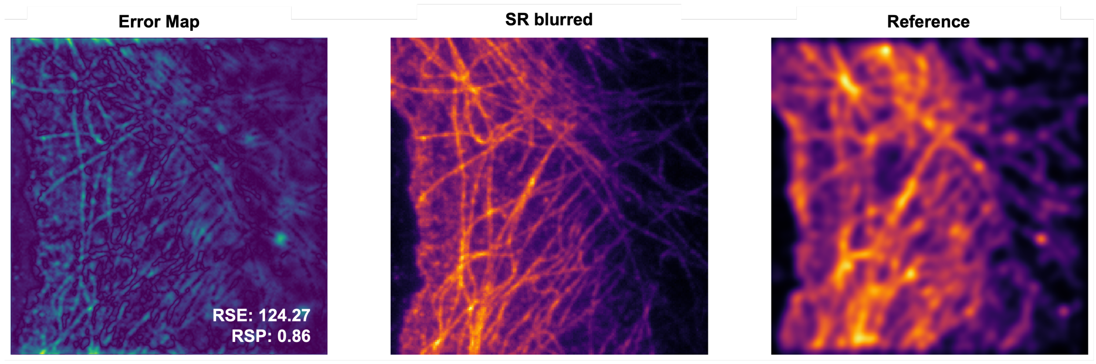


**Figure 4.8:** Example of an error map (left), the corresponding blurred super-resolved image (middle) and the reference TIRF image (right). The error map is obtained by comparing the blurred super-resolved image with the obtained TIRF image.

To streamline the process of calculating the error map and resolution metrics, a Python script was developed within the NanoPyx platform, facilitating the automation of plotting the relevant metrics for each ratio. The mean and standard deviation of the RSE, RSP and resolution were calculated for each density, and the results are presented in Figure 4.9.

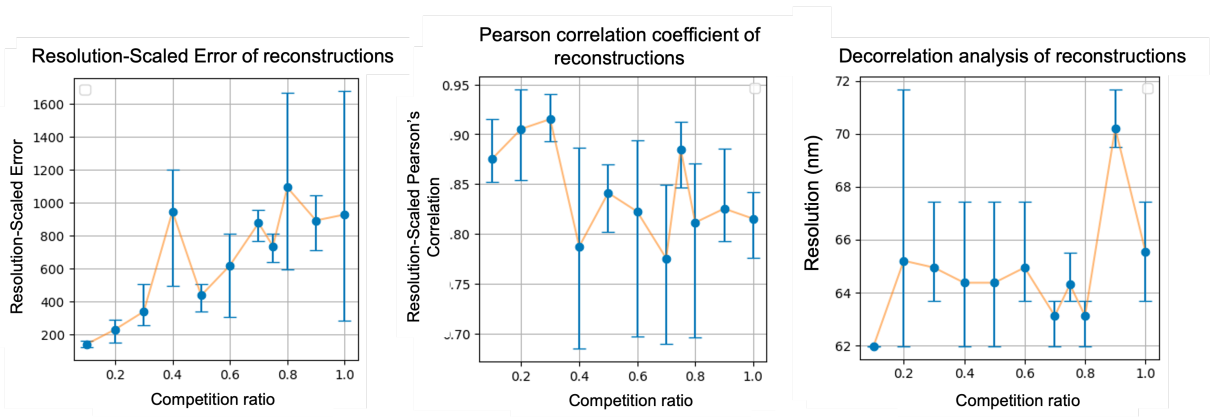


**Figure 4.9:** Mean RSE (left), RSP (middle) and resolution (right) for each paired competition ratio. The error bars represent the standard deviation.

From the results obtained from the mean RSE and RSP metrics (Figure 4.9, left and middle plots), extracting precise and definitive quantitative information from these metrics was challenging. This is due to the presence of very high standard deviations of this metrics in the acquired data.

Despite the variability introduced by the standard deviations, it is possible to discern a subtle pattern when examining the trends of RSE and RSP values. With an increase in the labelled/unlabelled ratio, a slight upward trend in the RSE is observed, along with a simultaneous downward trend in the RSP. This observation suggests that as the labelling density increases, the potential for artifact creation also rises, leading to a decrease in the precision of the super-resolved reconstructions.

Upon observing the resolution obtained from the decorrelation analysis (Figure 4.9, right plot), the same challenge is present - the presence of exceptionally high standard deviations. This factor hampers the ability to draw reliable quantitative conclusions from the data.

## 4.4 Discussion and Future Perspectives

The results from the competitive labelling experiments suggest that using the paired competition ratio could be an effective method of experimentally controlling the number of labelled molecules. The key supporting evidence for this lies in the plot showing the number of detections over the labelled/ unlabelled ratio (Figure 4.7). Although it is not possible to definitively ascertain whether the real molecular count is linearly controlled by the paired competition ratio experimentally (as the real number of labelled molecules cannot be measured), the observed shape of the plot aligns with the obtained results through the simulations (see zommed-in region of Figure 3.2): a linear increase in the number of localisations with an increasing ratio, followed by a plateau and a subsequent decrease in detections. This implies that controlling the number of labelled molecules experimentally was successful.

However, it is essential to acknowledge certain unexpected nuances that might contribute to the variability observed in the number of detections (SNR) between experiments. Factors such as natural cell-to-cell variability and uncontrollable experimental conditions could introduce variations. For instance, the use of TIRF imaging exposes microtubules closer to the coverslip to more illumination, leading to potential differences in the number of detections within the same field of view. Additionally, the laser illumination is higher at the centre of the field of view, resulting in potentially varying levels of illumination for microtubules in the central versus peripheral regions. These factors could account for the variability observed in the results, clearly observed in Tables A.1 and A.2.

The competitive labelling technique employed to control molecular counts may introduce some potential limitations and important biochemical considerations when combining labelled and unlabelled antibodies. One crucial assumption is that both labelled and unlabelled antibodies exhibit the same affinity for the tubulin epitope, implying equal binding probabilities. However, in reality, their interactions might not be identical, as they could have different binding characteristics. This discrepancy in binding interactions could lead to variations in the labelling efficiency and detection of molecules. If the labelled and unlabelled antibodies differ in their binding affinities, it may affect the competition dynamics between them. This, in turn, could result in unexpected variations in the number of labelled molecules detected, influencing the overall reliability and accuracy of the results. Therefore, while the paired competition ratio method provides a useful strategy of controlling the number of labelled molecules, researchers should be mindful of these potential shortcomings and the complexities arising from antibody-antigen interactions. Considering these factors in the experimental design by adjusting the concentrations of the antibodies according to the binding probabilities of both antibodies could help mitigate these issues.

The results obtained from the error map metrics (RSE and RSP) did not yield clear and quantitatively meaningful outcomes due to high standard deviations, making it challenging to discern any consistent pattern. Several factors could contribute to this outcome. Firstly, the error map is sensitive to false positive localisations, which can occur randomly and affect the results. Moreover, in the more undersampled regions with low density, the error map might assign greater significance to noise, as the SNR was poor. Additionally, and more importantly, the variability observed from cell to cell can significantly influence the analysis. Unlike simulations where conditions are precisely controlled, the uncontrolled factors involved in these experiments can contribute to the error map variations. To address this issue in future studies, one potential approach could involve using a very small fraction of the FOV for the error map analysis. However, implementing this method introduces subjectivity, as researchers may choose regions with or without merging artifacts based on local density. Balancing the need for more localised analysis while avoiding subjective bias remains a challenge that requires careful consideration in future studies.

Another aspect of the analysis involved the resolution assessment using the decorrelation analysis. However, similar to the error map metrics, the obtained results also presented high fluctuations in resolution for the same ratio but different cells. This variability is likely attributed to the same reasons as observed in the error map analysis. The decorrelation analysis, like the error map metrics, is sensitive to uncontrolled variability between cells, making it challenging to draw meaningful conclusions. The original intention behind using the decorrelation analysis as an additional metric was to gain insight into whether the microtubule structure was being accurately sampled. The hypothesis was that a decrease in the number of labelled molecules might lead to a deterioration in resolution due to undersampling. However, this interpretation might not hold true because the decorrelation analysis calculates resolution using the entire image and involves blurring. As a result, it does not provide a localised assessment of resolution, which is crucial when dealing with varying labelling densities in different regions of the sample. Hence, this limitation might explain the lack of informative results obtained from the decorrelation analysis.

The key insight gained from this study is the ability to assess the performance of the localisation algorithms using real experimental data. By understanding how the algorithms behave under different labelling densities, one can determine the optimal point at which accurate molecule counting is feasible. The investigation indicates that the WLS fitting algorithm is in a linear detection regime up to a 20% labelling ratio of molecules. However, at this density, there is a noticeable visual undersampling of the microtubule structure. Furthermore, the detection plot's slope diminishes gradually until 80% of labelled molecules, suggesting a degree of undercounting and consequent merging artifact creation by the algorithm. Yet, it still counts new molecules until the 80% labeling threshold, and no visible merging artifacts or undersampling are present, which suggests that this point might represent an optimal trade-off equilibrium. This understanding is useful to optimise imaging conditions to achieve high-quality results in a shorter time frame. By biochemically introducing a lower density of labelled molecules, the stringent imaging conditions required to achieve low labelling densities per frame can be eased. This means using lower laser powers and acquiring fewer frames, making the imaging process more efficient. This optimisation is particularly relevant for high throughput imaging scenarios, where a large number of images need to be acquired and analysed, and computational costs can be significant.

An important limitation of this methodology and analysis lies in the difficulty of quantifying the undersampling of biological structures. In this study, microtubules were used as the biological structure, and their continuous nature allowed to easily infer whether they were accurately sampled or undersampled. However, most biological structures of interest in SMLM are unknown and challenging to assess. For instance, when investigating protein aggregates, which typically consist of a small number of molecules, determining whether they are undersampled in super-resolved images becomes intricate. This raises the question of whether the competitive labelling approach is applicable to other structures and how it can be leveraged effectively.

In theory, the competitive labelling method could be extended to diverse biological structures, since the primary antibodies would be the only component that needed be altered. This means that it should be possible to gain insights into whether the localisation algorithm is accurately counting molecules. However, even if a plateau is identified in the number of detections by the algorithm, and it is still operating in a linear regime with the number of labels, it is not possible to determine if the structure is being undersampled. The reason is the lack of information from the unlabelled antibodies in the structure.

To potentially address this limitation, a future improvement to this method would be to use two different competing labels instead of one labelled and one unlabelled antibody. By using two colours, it should be possible to extract information from one colour (in one channel) and cross-reference it with the other channel, potentially revealing if the biological structure is being undersampled. However, finding two fluorophores that exhibit optimal blinking behavior in the same buffer is not a trivial task. Most dyes suited for SMLM are often in the far-red spectra (e.g., AF647, CF680), whose emission spectra overlap, complicating the unambiguous separation of information from each dye. One potential solution involves employing spectral demixing, a technique that can effectively disentangle information from two dyes with overlapping emission spectra. Although this approach introduces additional complexity, it might provide valuable insights into finding the optimal trade-off between the creation of merging artifacts by the localisation algorithm and the undersampling of the structure, for a wider range of biological structures.

Another potential improvement to this methodology involves exploring the performance of various localisation algorithms under different labelling densities. As demonstrated in the simulations presented in the previous chapter, the labelling densities can affect the behaviour of these algorithms differently. In this study, it was used the WLS with an Integrated Gaussian PSFfitting. However, investigating other algorithms, such as the MLE and the multi-emitter fitting approach, could offer valuable insights into their effectiveness at counting molecules in different labelling densities for experimental setups.

To conclude, the experimental tools and analysis methods developed in this study represent initial efforts towards accurately estimating the ability of algorithms to count molecules in real experimental data, enhancing the throughput and efficiency of SMLM imaging and analysis.

# 5 Conclusion

Super-resolution fluorescence microscopy techniques such as SMLM have become indispensable tools for cell biology, enabling researchers to visualise nanoscale cellular organisation and dynamics with high molecular specificity. However, reaching the full potential of these methods poses both computational and analytical challenges. This work strides towards overcoming key obstacles in single-molecule super-resolution imaging through advances in computational optimisation, simulation, and experimental design.

The NanoPyx framework introduced in Chapter 2 exemplifies an innovative approach to accelerate super-resolution data processing. Its core agent-based system, the Liquid Engine, dynamically selects optimal analysis implementations for each user device and input data. This adaptive optimisation enabled NanoPyx to achieve order-of-magnitude speedups for specific analysis tasks, promising to alleviate growing computational bottlenecks as super-resolution datasets expand. The adaptive performance strategy behind the Liquid Engine could inspire analogous advances in other domains facing rising computational demands.

Complementing these computational contributions, Chapters 3 and 4 address fundamental limitations in accurately resolving and counting molecules in SMLM. The Monte Carlo particle simulator devised in Chapter 3 offers versatile applications for investigating SMLM systems. Simulations with specific parameters were used to systematically assess SMLM localisation algorithm's performance under controlled conditions. Using this framework, an upper limit of $\sim$1 molecule/$\mu$m$^2$ for reliable particle counting was found employing standard Gaussian integral fitting approaches (MLE and WLS).

Moreover, Chapter 4 demonstrated a competitive labelling experimental strategy for SMLM algorithm characterisation. The detected molecular trend, in tune with the simulations, revealed the feasibility of controlling labelling densities in SMLM experiments. This system provided insights into a trade-off between inaccurate molecular counting by the localisation algorithms and undersampling of the biological structure. At 80% of labelled molecules, the point beyond which no additional molecules were detected, the reconstructions showed no visual merging artifacts or undersampling. Limitations in quantitatively confirming linearity and adequate sampling still remain. Overall, competitive labelling represents an important step towards characterising algorithm performance using real SMLM data across controlled density ranges. Ongoing refinement of the technique will enable more rigorous algorithm assessment and aid researchers in optimising workflows for their specific biological questions and imaging constraints.

In summary, this thesis advances single-molecule super-resolution imaging and analysis through interlinked progress in computational performance, simulation design, and experimental control. Translating these proof-of-concept achievements into robust tools and workflows will require continuous method development and system optimisation. Exciting frontiers remain open for exploration as researchers relentlessly seek to extract quantitative insights from data at ever-finer scales. Ultimately, this work provides a foundation for pushing the envelope of optical nanoscopy.

# Bibliography

[1] D.-E. Zacharioudaki, I. Fitilis, and M. Kotti, “Review of fluorescence spectroscopy in environmental quality applications,” *Molecules*, vol. 27, p. 4801, 7 2022.

[2] N. Gustafsson, “Enabling live-cell super-resolution microscopy by computational analysis and fluorescent probe design.” 2017. [Online]. Available: https://discovery.ucl.ac.uk/id/eprint/10025032/

[3] A. Alva, E. Brito-Alarcón, A. Linares, E. Torres-García, H. O. Hernández, R. Pinto-Cámara, D. Martínez, P. Hernández-Herrera, R. D’Antuono, C. Wood, and A. Guerrero, “Fluorescence fluctuation-based super-resolution microscopy: Basic concepts for an easy start,” *Journal of Microscopy*, 8 2022.

[4] M. Narayanan Nair, “Functionalization of epitaxial graphene by metal intercalation and molecules,” 09 2013.

[5] M. Orrit, T. Ha, and V. Sandoghdar, “Single-molecule optical spectroscopy,” *Chemical Society Reviews*, vol. 43, p. 973, 2014.

[6] H. Li and J. C. Vaughan, “Switchable fluorophores for single-molecule localization microscopy,” *Chemical Reviews*, vol. 118, pp. 9412–9454, 9 2018.

[7] J. Vangindertael, R. Camacho, W. Sempels, H. Mizuno, P. Dedecker, and K. P. F. Janssen, “An introduction to optical super-resolution microscopy for the adventurous biologist,” *Methods and Applications in Fluorescence*, vol. 6, p. 022003, 3 2018.

[8] G. Jacquemet, A. F. Carisey, H. Hamidi, R. Henriques, and C. Leterrier, “The cell biologist’s guide to super-resolution microscopy,” *Journal of Cell Science*, vol. 133, 6 2020.

[9] M. J. Sanderson, I. Smith, I. Parker, and M. D. Bootman, “Fluorescence microscopy,” *Cold Spring Harbor Protocols*, vol. 2014, p. pdb.top071795, 10 2014.

[10] A. Ettinger and T. Wittmann, *Fluorescence live cell imaging*, 2014, pp. 77–94.

[11] P. P. Mondal and A. Diaspro, *Fundamentals of Fluorescence Microscopy*. Springer Netherlands, 2014.

[12] K. N. Fish, "Total internal reflection fluorescence (tirf) microscopy," *Current Protocols in Cytometry*, vol. 50, 10 2009.

[13] E. M. Kudalkar, T. N. Davis, and C. L. Asbury, "Single-molecule total internal reflection fluorescence microscopy," *Cold Spring Harbor Protocols*, vol. 2016, p. pdb.top077800, 5 2016.

[14] M. Lelek, M. T. Gyparaki, G. Beliu, F. Schueder, J. Griffié, S. Manley, R. Jungmann, M. Sauer, M. Lakadamyali, and C. Zimmer, "Single-molecule localization microscopy," *Nature Reviews Methods Primers*, vol. 1, p. 39, 6 2021.

[15] A. Jimenez, K. Friedl, and C. Leterrier, "About samples, giving examples: Optimized single molecule localization microscopy," *Methods*, vol. 174, pp. 100–114, 3 2020.

[16] K. Prakash, B. Diederich, R. Heintzmann, and L. Schermelleh, "Super-resolution microscopy: a brief history and new avenues," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 380, 4 2022.

[17] C. S. Smith, N. Joseph, B. Rieger, and K. A. Lidke, "Fast, single-molecule localization that achieves theoretically minimum uncertainty," *Nature Methods*, vol. 7, pp. 373–375, 5 2010.

[18] R. E. Gordon, "Electron microscopy: A brief history and review of current clinical application," pp. 119–135, 2014.

[19] E. Betzig, G. H. Patterson, R. Sougrat, O. W. Lindwasser, S. Olenych, J. S. Bonifacino, M. W. Davidson, J. Lippincott-Schwartz, and H. F. Hess, "Imaging intracellular fluorescent proteins at nanometer resolution," *Science*, vol. 313, pp. 1642–1645, 9 2006.

[20] S. T. Hess, T. P. Girirajan, and M. D. Mason, "Ultra-high resolution imaging by fluorescence photoactivation localization microscopy," *Biophysical Journal*, vol. 91, pp. 4258–4272, 12 2006.

[21] M. J. Rust, M. Bates, and X. Zhuang, "Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm)," *Nature Methods*, vol. 3, pp. 793–796, 10 2006.

[22] S. van de Linde, A. Löschberger, T. Klein, M. Heidbreder, S. Wolter, M. Heilemann, and M. Sauer, "Direct stochastic optical reconstruction microscopy with standard fluorescent probes," *Nature Protocols*, vol. 6, pp. 991–1009, 7 2011.

[23] M. Bates, B. Huang, G. T. Dempsey, and X. Zhuang, "Multicolor super-resolution imaging with photo-switchable fluorescent probes," *Science*, vol. 317, pp. 1749–1753, 9 2007.

[24] R. Diekmann, M. Kahnwald, A. Schoenit, J. Deschamps, U. Matti, and J. Ries, "Optimizing imaging speed and excitation intensity for single-molecule localization microscopy," *Nature Methods*, vol. 17, pp. 909–912, 9 2020.

[25] R. R. Neubig, M. Spedding, T. Kenakin, and A. Christopoulos, "International union of pharmacology committee on receptor nomenclature and drug classification. xxxviii. update on terms and symbols in quantitative pharmacology," *Pharmacological Reviews*, vol. 55, pp. 597–606, 12 2003.

[26] G. Cooper, "Microtubules," 2000. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK9932/

[27] T. Scientific, "Alexa fluor 647 dye." [Online]. Available: https://www.thermofisher.com/pt/en/home/life-science/cell-analysis/fluorophores/alexa-fluor-647.html

[28] N. O. V. A. Hadjer Boukhatem, "An optimized buffer for repeatable multicolor storm," 2022. [Online]. Available: https://doi.org/10.1101/2022.05.19.491818

[29] S. van de Linde, S. Wolter, M. Heilemann, and M. Sauer, "The effect of photoswitching kinetics and labeling densities on super-resolution fluorescence imaging," *Journal of Biotechnology*, vol. 149, pp. 260–266, 9 2010.

[30] J. Xu, H. Ma, and Y. Liu, "Stochastic optical reconstruction microscopy (storm)," *Current Protocols in Cytometry*, vol. 81, 7 2017.

[31] B. Zhang, J. Zerubia, and J.-C. Olivo-Marin, "Gaussian approximations of fluorescence microscope point-spread function models," *Applied Optics*, vol. 46, p. 1819, 4 2007.

[32] F. Huang, S. L. Schwartz, J. M. Byars, and K. A. Lidke, "Simultaneous multiple-emitter fitting for single molecule super-resolution imaging," *Biomedical Optics Express*, vol. 2, p. 1377, 5 2011.

[33] L. C. Andrews, *Special functions of mathematics for engineers*. SPIE Press, 1998, iSBN: 9780819426161.

[34] D. Sage, H. Kirshner, T. Pengo, N. Stuurman, J. Min, S. Manley, and M. Unser, "Quantitative evaluation of software packages for single-molecule localization microscopy," *Nature Methods*, vol. 12, pp. 717–724, 8 2015.

[35] K. I. Mortensen, L. S. Churchman, J. A. Spudich, and H. Flyvbjerg, "Optimized localization analysis for single-molecule tracking and super-resolution microscopy," *Nature Methods*, vol. 7, pp. 377–381, 5 2010.

[36] Y.-L. Wu, P. Hoess, A. Tschanz, U. Matti, M. Mund, and J. Ries, "Maximum-likelihood model fitting for quantitative analysis of smlm data," *Nature Methods*, vol. 20, pp. 139–148, 1 2023.

[37] M. Galrinho, C. Rojas, and H. Hjalmarsson, “A weighted least-squares method for parameter estimation in structured models,” in *53rd IEEE Conference on Decision and Control*, 2014, pp. 3322–3327.

[38] L. Li, B. Xin, W. Kuang, Z. Zhou, and Z.-L. Huang, “Divide and conquer: real-time maximum likelihood fitting of multiple emitters for super-resolution localization microscopy,” *Optics Express*, vol. 27, p. 21029, 7 2019.

[39] M. Ovesný, P. Křížek, J. Borkovec, Z. Švindrych, and G. M. Hagen, “Thunderstorm: a comprehensive imagej plug-in for palm and storm data analysis and super-resolution imaging,” *Bioinformatics*, vol. 30, pp. 2389–2390, 8 2014.

[40] J. Schindelin, C. T. Rueden, M. C. Hiner, and K. W. Eliceiri, “The imagej ecosystem: An open platform for biomedical image analysis,” *Molecular Reproduction and Development*, vol. 82, pp. 518–529, 7 2015.

[41] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona, “Fiji: an open-source platform for biological-image analysis,” *Nature Methods*, vol. 9, pp. 676–682, 7 2012.

[42] H. Mazidi, T. Ding, A. Nehorai, and M. D. Lew, “Quantifying accuracy and heterogeneity in single-molecule super-resolution microscopy,” *Nature Communications*, vol. 11, p. 6353, 12 2020.

[43] K. J. A. Martens, B. Turkowyd, and U. Endesfelder, “Raw data to results: A hands-on introduction and overview of computational analysis for single-molecule localization microscopy,” *Frontiers in Bioinformatics*, vol. 1, 2 2022.

[44] K. N. Richter, N. H. Revelo, K. J. Seitz, M. S. Helm, D. Sarkar, R. S. Saleeb, E. D’Este, J. Eberle, E. Wagner, C. Vogl, D. F. Lazaro, F. Richter, J. Coy-Vergara, G. Coceano, E. S. Boyden, R. R. Duncan, S. W. Hell, M. A. Lauterbach, S. E. Lehnart, T. Moser, T. F. Outeiro, P. Rehling, B. Schwappach, I. Testa, B. Zapiec, and S. O. Rizzoli, “Glyoxal as an alternative fixative to formaldehyde in immunostaining and super-resolution microscopy,” *The EMBO Journal*, vol. 37, pp. 139–159, 1 2018.

[45] Y. K. Kim, J. Lee, X. Bi, H. Ha, S. H. Ng, Y. Ahn, J. Lee, B. K. Wagner, P. A. Clemons, and Y. Chang, “The binding of fluorophores to proteins depends on the cellular environment,” *Angewandte Chemie International Edition*, vol. 50, pp. 2761–2763, 3 2011.

[46] J. C. Waters, “Accuracy and precision in quantitative fluorescence microscopy,” *The Journal of Experimental Medicine*, vol. 206, pp. i15–i15, 7 2009.

[47] T. J. Lambert and J. C. Waters, “Navigating challenges in the application of superresolution microscopy,” *Journal of Cell Biology*, vol. 216, pp. 53–63, 1 2017.

[48] P. Fox-Roberts, R. Marsh, K. Pfisterer, A. Jayo, M. Parsons, and S. Cox, “Local dimensionality determines imaging speed in localization microscopy,” *Nature Communications*, vol. 8, p. 13558, 1 2017.

[49] F. Baumgart, A. M. Arnold, K. Leskovar, K. Staszek, M. Fölser, J. Weghuber, H. Stockinger, and G. J. Schütz, “Varying label density allows artifact-free analysis of membrane-protein nanoclusters,” *Nature Methods*, vol. 13, pp. 661–664, 8 2016.

[50] K. Kikuchi, L. D. Adair, J. Lin, E. J. New, and A. Kaur, “Photochemical mechanisms of fluorophores employed in single-molecule localization microscopy,” *Angewandte Chemie International Edition*, vol. 62, 1 2023.

[51] L. Patel, N. Gustafsson, Y. Lin, R. Ober, R. Henriques, and E. Cohen, “A hidden markov model approach to characterizing the photo-switching behavior of fluorophores,” *The Annals of Applied Statistics*, vol. 13, 9 2019.

[52] T. F. Scientific, “Alexa fluor 568 dye,” https://www.thermofisher.com/us/en/home/life-science/cell-analysis/fluorophores/alexa-fluor-568.html, 2023, accessed on August 23, 2023.

[53] Biotium, “Cf®680 and cf®680r dyes,” https://biotium.com/technology/cf-dyes/cf680-and-cf680r-dyes/, Unknown, accessed on August 23, 2023.

[54] A.-T. GmbH, “Product information: Atto 488,” https://www.atto-tec.com/fileadmin/user_upload/Katalog_Flyer_Support/ATTO_488.pdf, 2022, accessed on August 23, 2023.

[55] L. Nahidiazar, A. V. Agronskaia, J. Broertjes, B. van den Broek, and K. Jalink, “Optimizing imaging conditions for demanding multi-color super resolution localization microscopy,” *PLOS ONE*, vol. 11, p. e0158884, 7 2016.

[56] H. Ma and Y. Liu, “Super-resolution localization microscopy: Toward high throughput, high quality, and low cost,” *APL Photonics*, vol. 5, 6 2020.

[57] P. Nakashima and A. Johnson, “Measuring the psf from aperture images of arbitrary shape—an algorithm,” *Ultramicroscopy*, vol. 94, pp. 135–148, 2 2003.

[58] U. Endesfelder and M. Heilemann, “Art and artifacts in single-molecule localization microscopy: beyond attractive images,” *Nature Methods*, vol. 11, pp. 235–238, 3 2014.

[59] R. E. Thompson, D. R. Larson, and W. W. Webb, “Precise nanometer localization analysis for individual fluorescent probes,” *Biophysical Journal*, vol. 82, pp. 2775–2783, 5 2002.

[60] B. Rieger and S. Stallinga, “The lateral and axial localization uncertainty in super-resolution light microscopy,” *ChemPhysChem*, vol. 15, pp. 664–670, 3 2014.

[61] A. Burgert, S. Letschert, S. Doose, and M. Sauer, “Artifacts in single-molecule localization microscopy,” *Histochemistry and Cell Biology*, vol. 144, pp. 123–131, 8 2015.

[62] A. R. Small, “Theoretical limits on errors and acquisition rates in localizing switchable fluorophores,” *Biophysical Journal*, vol. 96, pp. L16–L18, 1 2009.

[63] A. Small and S. Stahlheber, “Fluorophore localization algorithms for super-resolution microscopy,” *Nature Methods*, vol. 11, pp. 267–279, 3 2014.

[64] N. Gustafsson, S. Culley, G. Ashdown, D. M. Owen, P. M. Pereira, and R. Henriques, “Fast live-cell conventional fluorophore nanoscopy with imagej through super-resolution radial fluctuations,” *Nature Communications*, vol. 7, p. 12471, 11 2016.

[65] J. Li, L. Yang, and Y. Zhong, “Image scaling based on the catmull-rom spline surfaces with free parameters.” Atlantis Press, 2018.

[66] H. Ma, F. Long, S. Zeng, and Z.-L. Huang, “Fast and precise algorithm based on maximum radial symmetry for single molecule localization,” *Optics Letters*, vol. 37, p. 2481, 7 2012.

[67] R. F. Laine, H. S. Heil, S. Coelho, J. Nixon-Abell, A. Jimenez, T. Galgani, A. Stubb, G. Follain, S. Culley, G. Jacquemet, B. Hajj, C. Leterrier, and R. Henriques, “High-fidelity 3d live-cell nanoscopy through data-driven enhanced super-resolution radial fluctuation,” *bioRxiv*, 2022. [Online]. Available: https://www.biorxiv.org/content/early/2022/04/08/2022.04.07.487490

[68] R. Bracewell, “The fast hartley transform,” *Proceedings of the IEEE*, vol. 72, no. 8, pp. 1010–1018, 1984.

[69] R. F. Laine, K. L. Tosheva, N. Gustafsson, R. D. M. Gray, P. Almada, D. Albrecht, G. T. Risa, F. Hurtig, A.-C. Lindås, B. Baum, J. Mercer, C. Leterrier, P. M. Pereira, S. Culley, and R. Henriques, “Nanoj: a high-performance open-source super-resolution microscopy toolbox,” *Journal of Physics D: Applied Physics*, vol. 52, p. 163001, 4 2019.

[70] J. W. Pylvänäinen, R. F. Laine, B. M. S. Saraiva, S. Ghimire, G. Follain, R. Henriques, and G. Jacquemet, “Fast4dreg – fast registration of 4d microscopy datasets,” *Journal of Cell Science*, vol. 136, 2 2023.

[71] S. Culley, D. Albrecht, C. Jacobs, P. M. Pereira, C. Leterrier, J. Mercer, and R. Henriques, “Quantitative mapping and minimization of super-resolution optical imaging artifacts,” *Nature Methods*, vol. 15, pp. 263–266, 4 2018.

[72] R. P. J. Nieuwenhuizen, K. A. Lidke, M. Bates, D. L. Puig, D. Grünwald, S. Stallinga, and B. Rieger, "Measuring image resolution in optical nanoscopy," *Nature Methods*, vol. 10, pp. 557–562, 6 2013.

[73] A. Descloux, K. S. Grußmayer, and A. Radenovic, "Parameter-free image resolution estimation based on decorrelation analysis," *Nature Methods*, vol. 16, pp. 918–924, 9 2019.

[74] J. E. Stone, D. Gohara, and G. Shi, "Opencl: A parallel programming standard for heterogeneous computing systems," *Computing in Science and Engineering*, vol. 12, pp. 66–73, 5 2010.

[75] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, "Cython: The best of both worlds," *Computing in Science and Engineering*, vol. 13, no. 2, pp. 31–39, 2011.

[76] A. et al. Kloeckner, "Pyopencl," 2022.

[77] S. K. Lam, A. Pitrou, and S. Seibert, "Numba." ACM, 11 2015, pp. 1–6.

[78] I. M. Cunha, B. M. Saraiva, A. D. Brito, G. Follain, R. Portela, R. Haase, P. M. Pereira, G. Jacquemet, and R. Henriques, "Nanopyx: super-fast bioimage analysis powered by adaptive machine learning," *BioRxiv*, 8 2023. [Online]. Available: https://doi.org/10.1101/2023.08.13.553080

[79] M. J. Rust, M. Bates, and X. Zhuang, "Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm)," *Nature Methods*, vol. 3, pp. 793–796, 10 2006.

[80] M. Bates, B. Huang, and X. Zhuang, "Super-resolution microscopy by nanoscale localization of photo-switchable fluorescent probes," *Current Opinion in Chemical Biology*, vol. 12, pp. 505–514, 10 2008.

[81] S. W. Hell and J. Wichmann, "Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy," *Optics Letters*, vol. 19, p. 780, 6 1994.

[82] J. M. Guerra, "Super-resolution through illumination by diffraction-born evanescent waves," *Applied Physics Letters*, vol. 66, pp. 3555–3557, 6 1995.

[83] M. Ovesný, P. Křížek, J. Borkovec, Z. Švindrych, and G. M. Hagen, "Thunderstorm: a comprehensive imagej plug-in for palm and storm data analysis and super-resolution imaging," *Bioinformatics*, vol. 30, pp. 2389–2390, 8 2014.

[84] J. Schnitzbauer, M. T. Strauss, T. Schlichthaerle, F. Schueder, and R. Jungmann, "Super-resolution microscopy with dna-paint," *Nature Protocols*, vol. 12, pp. 1198–1228, 6 2017.

[85] M. Müller, V. Mönkemöller, S. Hennig, W. Hübner, and T. Huser, "Open-source image reconstruction of super-resolution structured illumination microscopy data in imagej," *Nature Communications*, vol. 7, p. 10980, 3 2016.

[86] T. et al Kluyver, “Jupyter notebooks – a publishing format for reproducible computational workflows,” *Positioning and Power in Academic Publishing: Players, Agents and Agendas 87–90 (IOS Press, 2016)*.

[87] N. et al. Sofroniew, “napari: a multi-dimensional image viewer for python,” 2022.

[88] N. Gustafsson, S. Culley, G. Ashdown, D. M. Owen, P. M. Pereira, and R. Henriques, “Fast live-cell conventional fluorophore nanoscopy with imagej through super-resolution radial fluctuations,” *Nature Communications*, vol. 7, p. 12471, 11 2016.

[89] E. Catmull and R. Rom, *A CLASS OF LOCAL INTERPOLATING SPLINES*. Elsevier, 1974, pp. 317–326.

[90] N. Oliver and D. Keller, “Storm vectashield datasets (tubulin),” *2023*.

[91] K. Czarnecki, K. Østerbye, and M. Völter, “Generative programming.” 01 2002, pp. 15–29.

[92] V. Novák, I. Perfilieva, and J. Močkoř, *Mathematical Principles of Fuzzy Logic*. Springer US, 1999.

[93] “Transonic: Make your python code fly at transonic speeds!” 2023, https://github.com/fluiddyn/transonic.

[94] R. Haase, L. A. Royer, P. Steinbach, D. Schmidt, A. Dibrov, U. Schmidt, M. Weigert, N. Maghelli, P. Tomancak, F. Jug, and E. W. Myers, “Clij: Gpu-accelerated image processing for everyone,” *Nature Methods*, vol. 17, pp. 5–6, 1 2020.

[95] S. Wolter, U. Endesfelder, S. van de Linde, M. Heilemann, and M. Sauer, “Measuring localization performance of super-resolution algorithms on very active samples,” *Optics Express*, vol. 19, p. 7020, 4 2011.

[96] A. V. Abraham, S. Ram, J. Chao, E. S. Ward, and R. J. Ober, “Quantitative study of single molecule location estimation techniques,” *Optics Express*, vol. 17, p. 23352, 12 2009.

[97] D. Bourgeois, “Single molecule imaging simulations with advanced fluorophore photophysics,” *Communications Biology*, vol. 6, p. 53, 1 2023.

[98] D. Sage, T.-A. Pham, H. Babcock, T. Lukes, T. Pengo, J. Chao, R. Velmurugan, A. Herbert, A. Agrawal, S. Colabrese, A. Wheeler, A. Archetti, B. Rieger, R. Ober, G. M. Hagen, J.-B. Sibarita, J. Ries, R. Henriques, M. Unser, and S. Holden, “Super-resolution fight club: assessment of 2d and 3d single-molecule localization microscopy software,” *Nature Methods*, vol. 16, pp. 387–395, 5 2019.

[99] V. Venkataramani, F. Herrmannsdörfer, M. Heilemann, and T. Kuner, “Suresim: simulating localization microscopy experiments from ground truth models,” *Nature Methods*, vol. 13, pp. 319–321, 4 2016.

[100] M. Lindén, V. Ćurić, A. Boucharin, D. Fange, and J. Elf, “Simulated single molecule microscopy with smeagol,” *Bioinformatics*, vol. 32, pp. 2394–2395, 8 2016.

[101] T. Novák, T. Gajdos, J. Sinkó, G. Szabó, and M. Erdélyi, “Teststorm: Versatile simulator software for multimodal super-resolution localization fluorescence microscopy,” *Scientific Reports*, vol. 7, p. 951, 4 2017.

[102] M. Lagardère, I. Chamma, E. Bouilhol, M. Nikolski, and O. Thoumine, “Fluosim: simulator of single molecule dynamics for fluorescence live-cell and super-resolution imaging of membrane proteins,” *Scientific Reports*, vol. 10, p. 19954, 11 2020.

[103] N. Metropolis and S. Ulam, “The monte carlo method,” *Journal of the American Statistical Association*, vol. 44, p. 335, 9 1949.

[104] W.-H. Yeo, Y. Zhang, A. E. Neely, X. Bao, C. Sun, and H. F. Zhang, “Investigating uncertainties in single-molecule localization microscopy using experimentally informed monte carlo simulation,” *Nano Letters*, 7 2023.

[105] R. A. Kerr, T. M. Bartol, B. Kaminsky, M. Dittrich, J.-C. J. Chang, S. B. Baden, T. J. Sejnowski, and J. R. Stiles, “Fast monte carlo simulation methods for biological reaction-diffusion systems in solution and on surfaces,” *SIAM Journal on Scientific Computing*, vol. 30, pp. 3126–3149, 1 2008.

[106] F. Huang, S. L. Schwartz, J. M. Byars, and K. A. Lidke, “Simultaneous multiple-emitter fitting for single molecule super-resolution imaging,” *Biomedical Optics Express*, vol. 2, p. 1377, 5 2011.

[107] C. T. Rueden, M. C. Hiner, E. L. Evans, M. A. Pinkert, A. M. Lucas, A. E. Carpenter, B. A. Cimini, and K. W. Eliceiri, “Pyimagej: A library for integrating imagej and python,” *Nature Methods*, vol. 19, pp. 1326–1327, 11 2022.

[108] J. Sinkó, R. Kákonyi, E. Rees, D. Metcalf, A. E. Knight, C. F. Kaminski, G. Szabó, and M. Erdélyi, “Teststorm: Simulator for optimizing sample labeling and image acquisition in localization based super-resolution microscopy,” *Biomedical Optics Express*, vol. 5, p. 778, 3 2014.

[109] D. F. Nino and J. N. Milstein, “Estimating the dynamic range of quantitative single-molecule localization microscopy,” *Biophysical Journal*, vol. 120, pp. 3901–3910, 9 2021.

[110] A. M. Bittel, I. Saldivar, N. Dolman, A. K. Nickerson, L.-J. Lin, X. Nan, and S. L. Gibbs, “Effect of labeling density and time post labeling on quality of antibody-based super resolution microscopy

images," J. Enderlein, I. Gregor, Z. K. Gryczynski, R. Erdmann, and F. Koberling, Eds., 3 2015, p. 93310M.

[111] A. M. Bittel, A. Nickerson, I. S. Saldivar, N. J. Dolman, X. Nan, and S. L. Gibbs, "Methodology for quantitative characterization of fluorophore photoswitching to predict superresolution microscopy image quality," *Scientific Reports*, vol. 6, p. 29687, 7 2016.

[112] M. Štefko, B. Ottino, K. M. Douglass, and S. Manley, "Autonomous illumination control for localization microscopy," *Optics Express*, vol. 26, p. 30882, 11 2018.

[113] G. Kulaitis, A. Munk, and F. Werner, "What is resolution? a statistical minimax testing perspective on super-resolution microscopy," *arxiv*, 5 2020.

# A

# Appendix

## A.1 eSRRF Algorithm

Given an input of a diffraction-limited image stack, the eSRRF method consists of a spatial analysis of each image frame, followed by a temporal analysis of the spatially transformed image stack.

Firstly, eSRRF performs a FHT interpolation to the diffraction-limited image stack to upsample the images, which minimises macro-pixel artifacts. Then, for a user defined area which contains the size of the PSF, and for each subpixel of that area, the intensity gradients in the x and y direction are calculated, and the subpixels are attributed a specific weight, in which subpixels further from the centre have an inferior weight than the ones at the centre of the pixel of interest. This allows a better estimation of the environment around the pixel of interest. The workflow of eSRRF can be seen in Figure A.1.

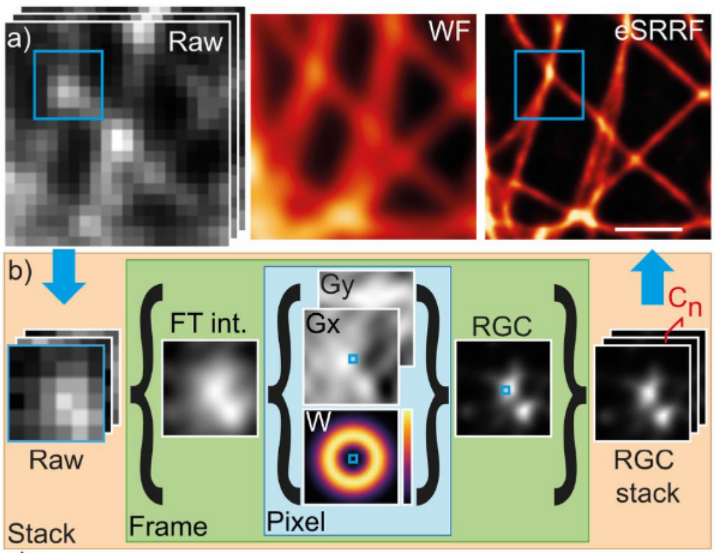


**Figure A.1: a)** Left: Acquired raw image stack. Middle: Obtained diffraction-limited image, via a wide-field microscope. Right: **eSSRF!** (**eSSRF!**) reconstruction. **b)** Workflow of **eSSRF!**. The selected diffraction-limited window from the raw image frames undergoes a Fourier Transform interpolation for upsampling (subpixelization). From the magnified window, the gradients in the horizontal and vertical directions and the weight map are calculated. Based on this, the RGC is calculated for each pixel to calculate the RGC map. Then, via temporal correlations, the RGC stack is compressed into a super-resolved image [67].

A more detailed explanation of the eSRRF algorithm (spatial and temporal analysis) can be found in the following sections.

### Spatial Transformation: Radial Gradient Convergence

The raw image is first interpolated spatially by a user-defined magnification factor $mag$, typically ranging from 2 to 5, using Fourier Interpolation via FHT. This reduces macro-pixel patterning effects.

The vertical and horizontal image gradients $G_x$ and $G_y$ are calculated using Roberts cross method:

$$G_a(i+0.5, j+0.5) = I(i,j) - I(i+1, j+1) \tag{A.1}$$

$$G_b(i+0.5, j+0.5) = I(i, j+1) - I(i+1, j) \tag{A.2}$$

Where $G_a$ and $G_b$ are the diagonal gradients. These are then rotated to align with image axes.

The pixel radius $R$ represents the PSF Full Width at Half Maximum (FWHM), so $\sigma = \frac{R}{2.355}$ estimates the PSF standard deviation [113].

For each pixel $(i_0, j_0)$, the RGC is computed over a local disk $\Delta$ of radius $2\sigma + 1$ as:

$$RGC(i_0, j_0) = \sum_{(i,j)\in\Delta} W(i_0, j_0, i, j) \cdot D_k(i_0, j_0, i, j) \tag{A.3}$$

Where the weighting factor $W$ based on Gaussian derivative is:

$$W(d) = \left[d \cdot e^{-\frac{d^2}{2\sigma^2}}\right]^4 \tag{A.4}$$

And the gradient convergence $D_k$ is:

$$D_k = 1 - \frac{|G_y(i,j) \cdot (i - i_0) - G_x(i,j) \cdot (j - j_0)|}{d\sqrt{G_x^2 + G_y^2}} \tag{A.5}$$

Where $d$ is the distance between $(i, j)$ and $(i_0, j_0)$.

**Temporal Analysis**

The RGC stacks are combined over time via temporal average (AVG), variance (VAR) or 2nd order autocumulant (TAC2).

$$AVG(i_0, j_0) = \langle RGC_t(i_0, j_0)\rangle_t \tag{A.6}$$

$$VAR(i_0, j_0) = \langle \delta RGC_t(i_0, j_0) \cdot \delta RGC_t(i_0, j_0)\rangle_t \tag{A.7}$$

$$TAC2(i_0, j_0) = \langle \delta RGC_t(i_0, j_0) \cdot \delta RGC_{t+\delta t}(i_0, j_0)\rangle_t \tag{A.8}$$

Where $\delta RGC_t = RGC_t - \langle RGC_t\rangle_t$

This exploits temporal fluctuations to further improve resolution and image fidelity.

## A.2 NanoPyx Supplementary Information

### A.2.1 NanoPyx Supplementary Figures

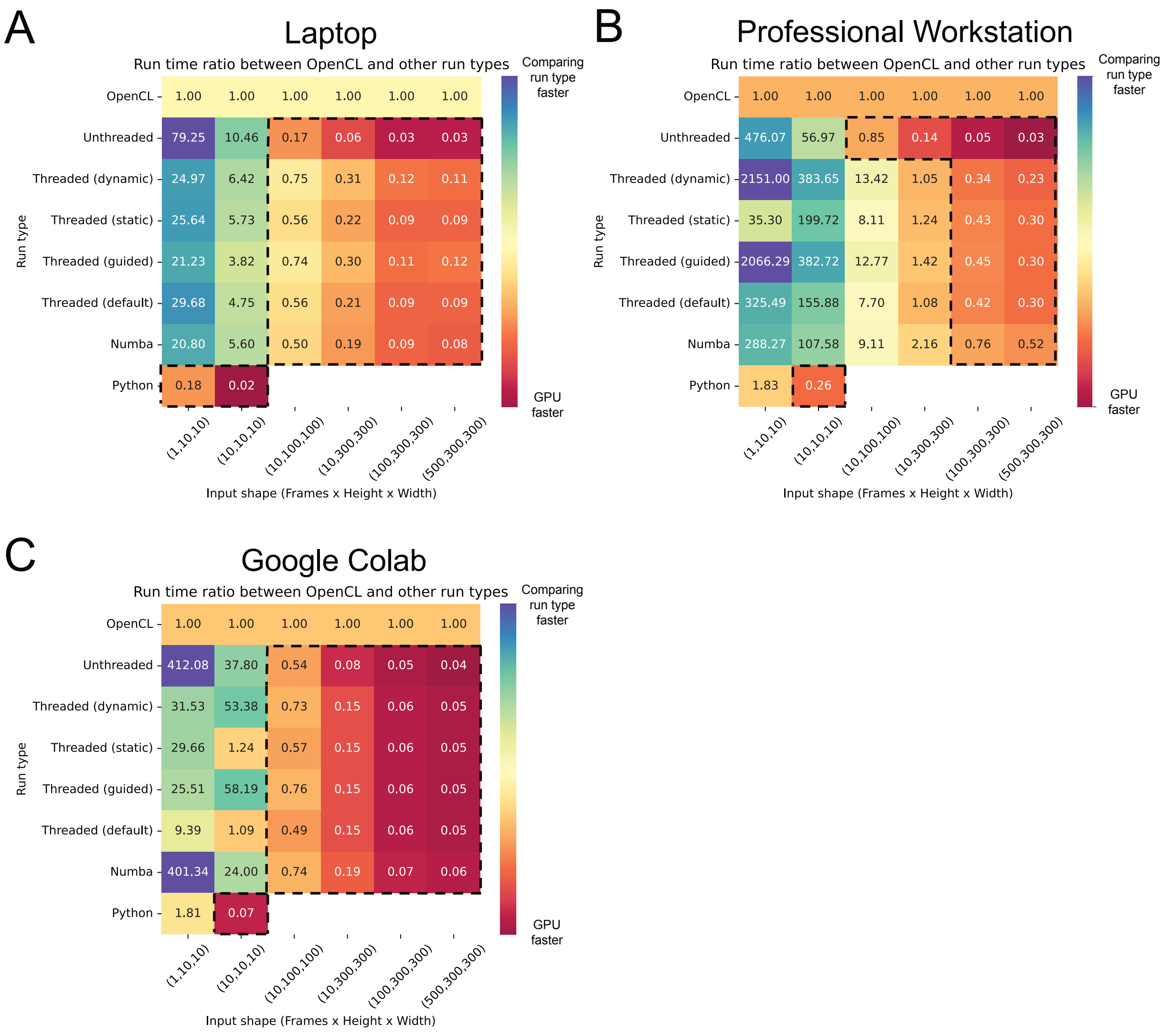


**Figure A.2: Ratio between the run times of OpenCL and other implemented run types.** Run times of a 5x Catmull-rom [89] interpolation were measured across multiple input data sizes using either a MacBook Air M1 (A), a Professional Workstation (B) or Google Collaboratory (C). Area within dashed lines correspond to kernel and image sizes where OpenCL is faster than other implementations.

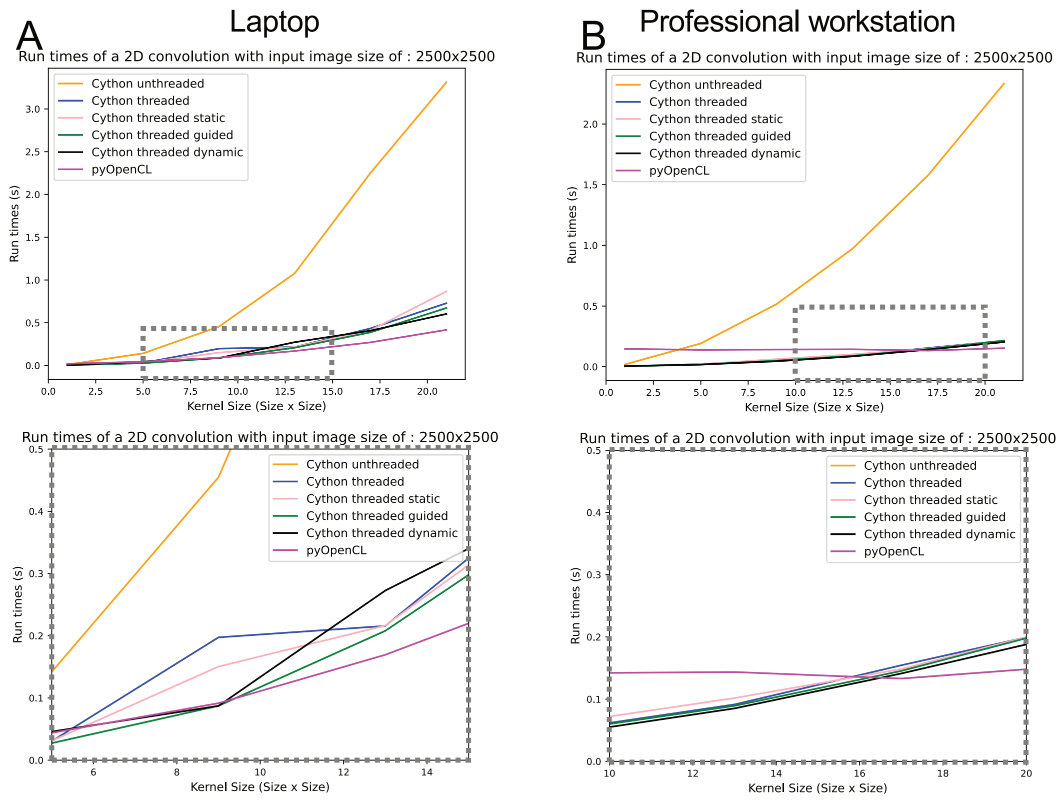


**Figure A.3: Kernel size impacts which implementation is the fastest.** A 2D convolution was performed on images with varying kernel sizes, ranging from 1 to 21 (every 4) using either a MacBook Air M1 (A) or a professional workstation (B). A 21 by 21 kernel was used in all operations. While unthreaded is virtually always the slowest implementation, the threaded implementations are only the fastest until the size increases to 20MB, after which PyOpenCL becomes the fastest. Bottom panels correspond to zoomed in windows of top panels, indicated by dotted boxes.

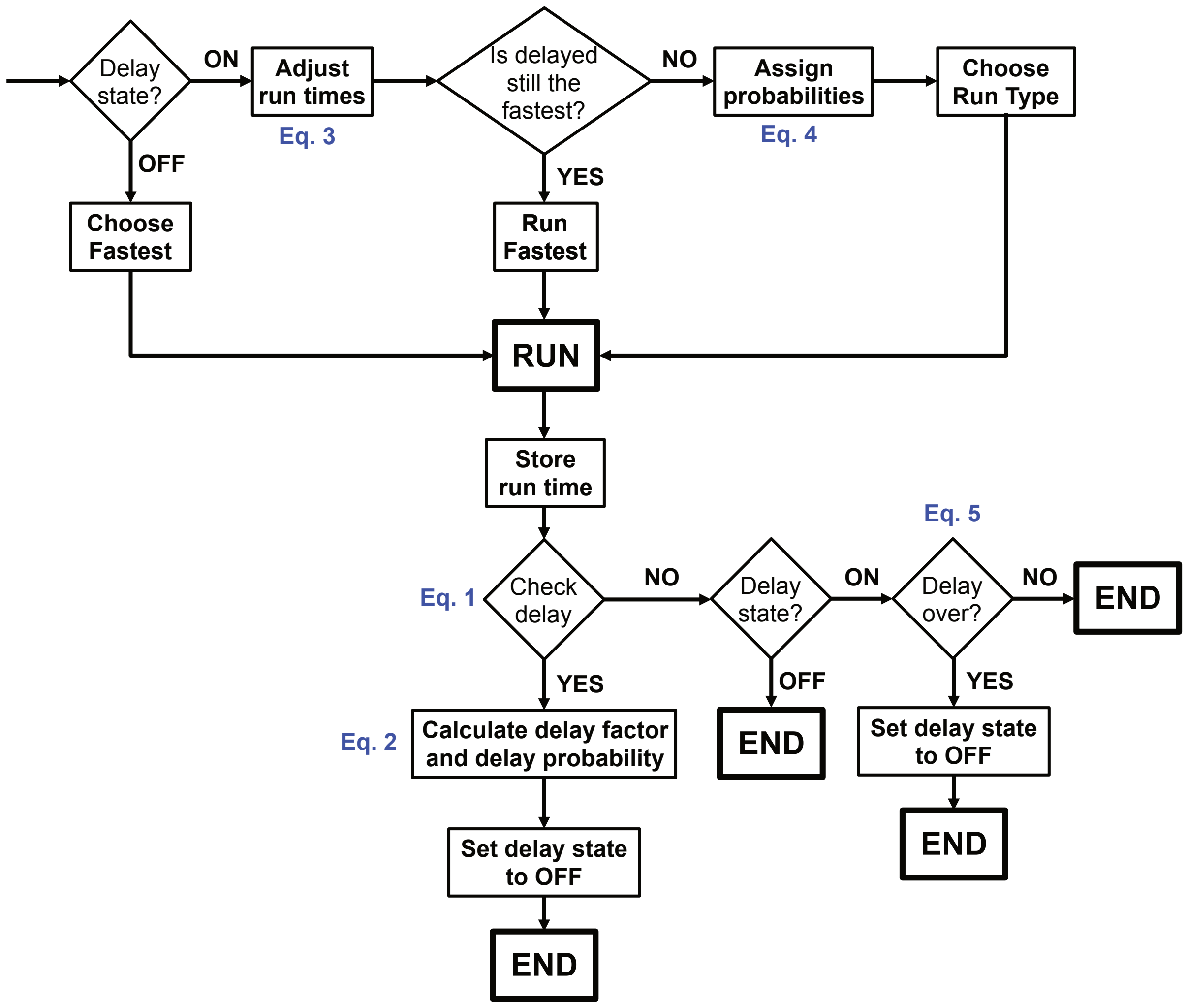
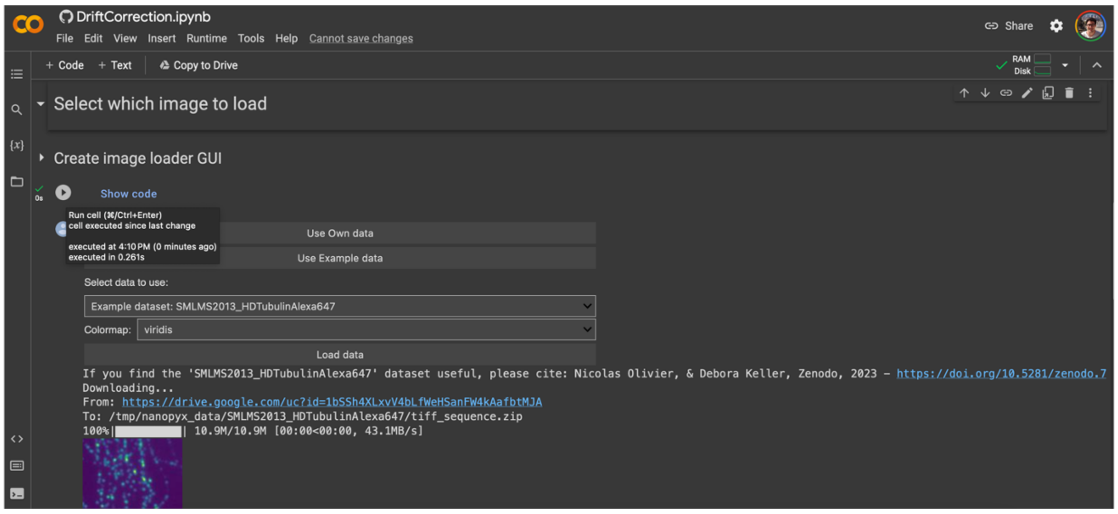


**Figure A.4:** Schematic of the agent decision making for delay management.

A



Select which image to load

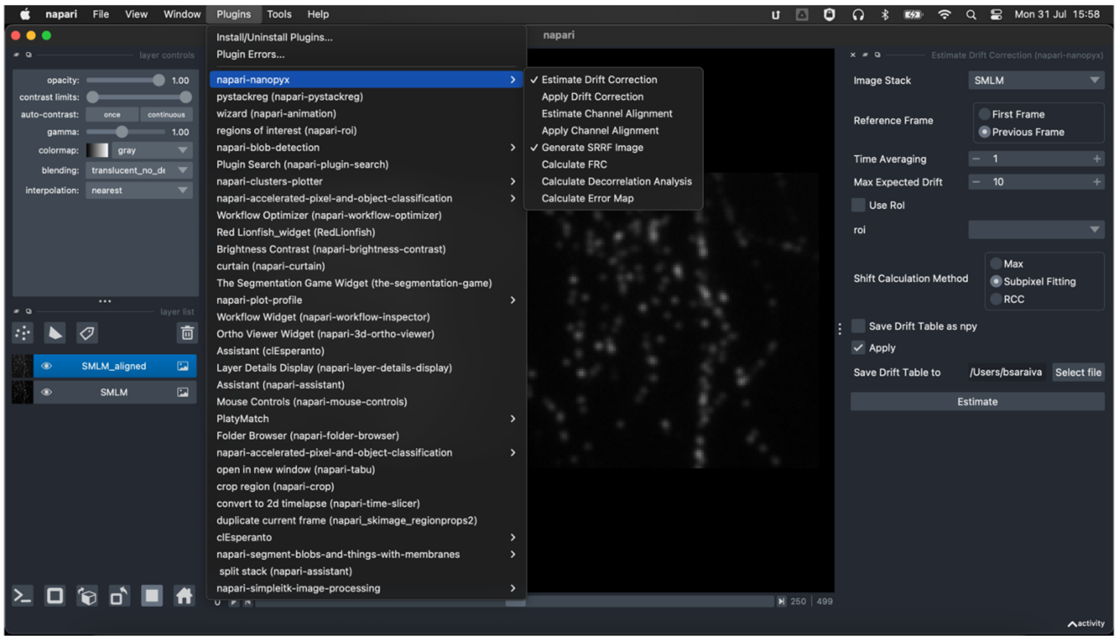
Create image loader GUI

Select data to use: Example dataset: SMLMS2013_HDTubulinAlexa647

Colormap: viridis

```
If you find the 'SMLMS2013_HDTubulinAlexa647' dataset useful, please cite: Nicolas Olivier, & Debora Keller, Zenodo, 2023 - https://doi.org/10.5281/zenodo.7
Downloading...
From: https://drive.google.com/uc?id=1bSSh4XLxvV4bLfWeHSanFW4kAafbtMJA
To: /tmp/nanopyx_data/SMLMS2013_HDTubulinAlexa647/tiff_sequence.zip
100%|██████████| 10.9M/10.9M [00:00<00:00, 43.1MB/s]
```

B


napari File View Window Plugins Tools Help
Mon 31 Jul 15:58
Install/Uninstall Plugins...
Plugin Errors...
napari-nanopyx
pystackreg (napari-pystackreg)
wizard (napari-animation)
regions of interest (napari-roi)
napari-blob-detection
Plugin Search (napari-plugin-search)
napari-clusters-plotter
napari-accelerated-pixel-and-object-classification
Workflow Optimizer (napari-workflow-optimizer)
Red Lionfish_widget (RedLionfish)
Brightness Contrast (napari-brightness-contrast)
curtain (napari-curtain)
The Segmentation Game Widget (the-segmentation-game)
napari-plot-profile
Workflow Widget (napari-workflow-inspector)
Ortho Viewer Widget (napari-3d-ortho-viewer)
Assistant (clEsperanto)
Layer Details Display (napari-layer-details-display)
Assistant (napari-assistant)
Mouse Controls (napari-mouse-controls)
PlatyMatch
Folder Browser (napari-folder-browser)
napari-accelerated-pixel-and-object-classification
open in new window (napari-tabu)
crop region (napari-crop)
convert to 2d timelapse (napari-time-slicer)
duplicate current frame (napari_skimage_regionprops2)
clEsperanto
napari-segment-blobs-and-things-with-membranes
split stack (napari-assistant)
napari-simpleitk-image-processing
✓ Estimate Drift Correction
Apply Drift Correction
Estimate Channel Alignment
Apply Channel Alignment
✓ Generate SRRF Image
Calculate FRC
Calculate Decorrelation Analysis
Calculate Error Map
napari
layer controls
layer list
SMLM_aligned
SMLM
250 499
activity
Select file
Estimate

opacity: 1.00
contrast limits:
auto-contrast: once continuous
gamma: 1.00
colormap: gray
blending: translucent_no_de
interpolation: nearest

Estimate Drift Correction (napari-nanopyx)

Image Stack: SMLM

Reference Frame: First Frame [✓ Previous Frame]

Time Averaging: 1

Max Expected Drift: 10

☐ Use RoI

roi:

Shift Calculation Method: Max [✓ Subpixel Fitting] RCC

☐ Save Drift Table as npy

✓ Apply

Save Drift Table to: /Users/bsaraiva

**Figure A.5: NanoPyx is available to users independently or their coding expertise.** Besides the using NanoPyx as a Python library, users also have access to Jupyter notebooks [86] (A) that can either be run locally or through Google Collaboratory and a napari [87] plugin (B).

### A.2.2 NanoPyx eSRRF implementation

As stated in section 1.1.5, eSRRF performs a spatial and temporal analysis of an input diffraction-limited stack, and outputs a super-resolved reconstruction. The full implementation of eSRRF into the Liquid Engine and its comparison against NanoJ-eSRRF is the focus of this section. The inputs of the eSRRF algorithm are a diffraction-limited image stack, a magnification factor, a radius in pixels, a sensitivity value and a boolean value on whether or not to perform intensity weighting. The following Figure A.6 shows the stepwise eSRRF workflow developed, given an input of an an artificially created PSF.

Figure A.6 provides clear evidence of the successful implementation of every stage in the workflow. The intensity gradient calculation's outcomes align with expectations: regions in the artificial PSF that correspond to edges exhibit higher values, mirroring the higher intensity variation in the initial image. Conversely, areas with similar intensities yield lower values in the gradient images. The interpolation steps are notably evident in their clear upsampling effect on the images. Moreover, the RGC maps corroborate anticipated results by showcasing a resolution enhancement relative to the original input. This is demonstrated by the emergence of a "smaller" PSF discernibly centered in the image, indicative of an improved resolving capability.

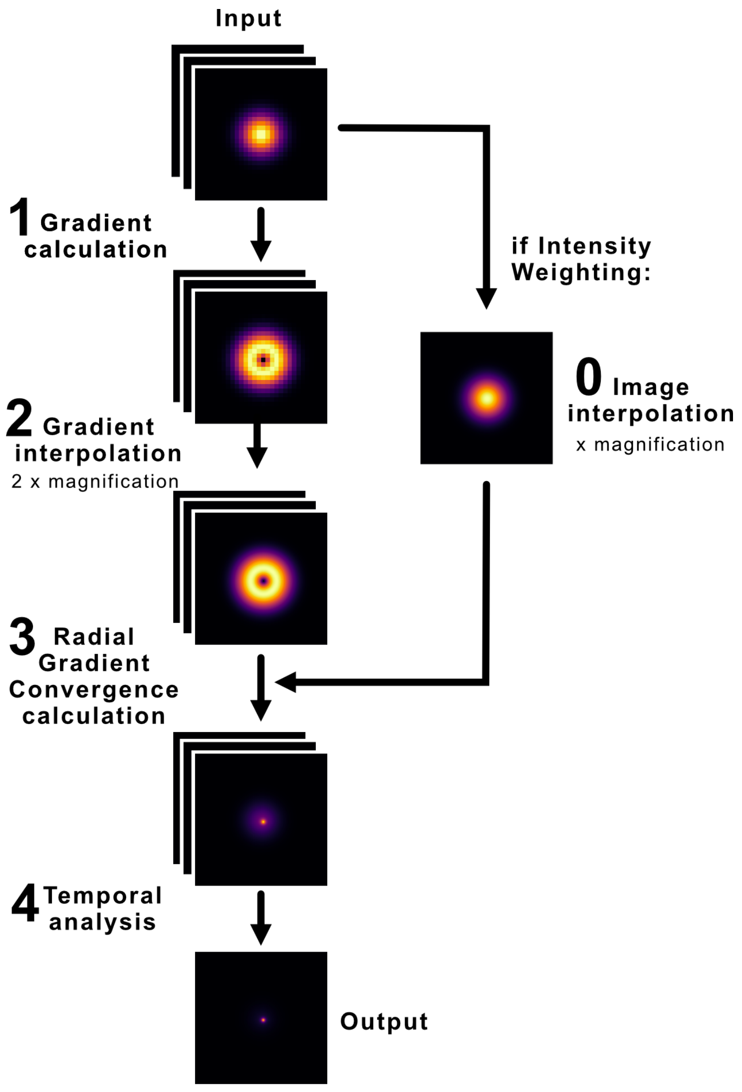


**Figure A.6:** Illustration of eSRRF workflow. Given an input of a diffraction-limited image stack, step **0** is a pre-interpolation step, where each frame is upsampled by a user-defined magnification factor (in this example, magnification = 5), if the boolean condition of doing intensity weighting is set to True. Using the original image stack, step **1** is to calculate the pixelwise intensity gradients in the horizontal and vertical direction of each frame of the stack. The illustrated gradient in the Figure is the gradient magnitude. Step **2** is to upsample the obtained horizontal and vertical gradients by a factor of 2 x the user-input magnification. Given the interpolated gradients in both directions, step **3** is then to calculate the Radial Gradient Convergence maps. Finally, step **4** is to calculate temporal correlations between the obtained maps to generate the final super-resolved image. For this example, it was used the mean between the frames.

For a more realistic assessment and for a direct comparison to the original method implemented in NanoJ, Figure A.7 shows the reconstruction of microtubules from the dataset [90] using NanoJ and NanoPyx. The same interpolation type (Catmull-Rom) and parameters were used.

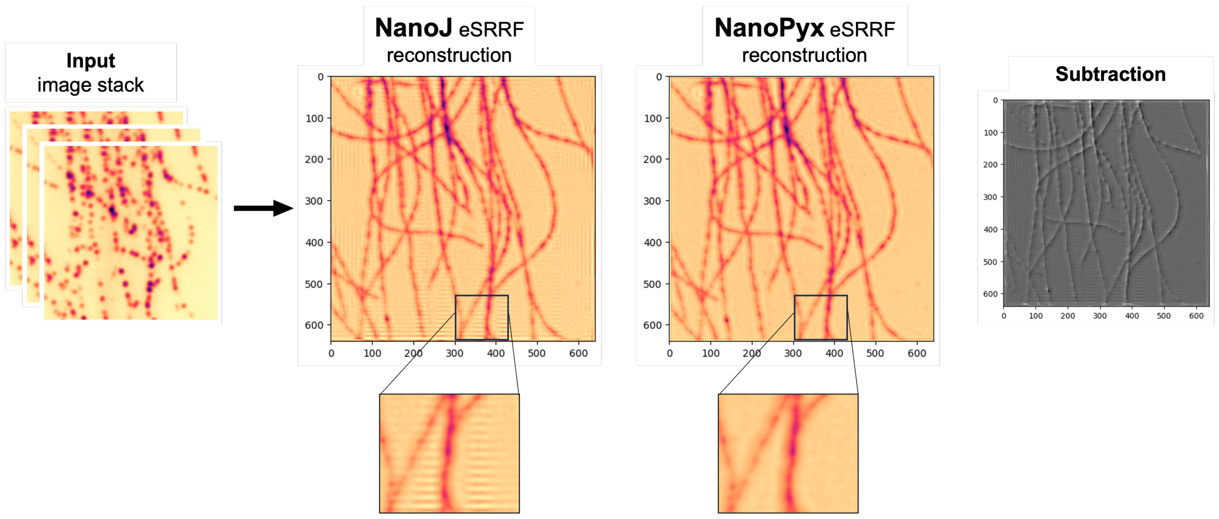


**Figure A.7:** Given the same input dataset (from [90]) (**left**), NanoJ and NanoPyx **(middle)** reconstructions were performed. Each reconstruction shows a zoomed-in region for easier visual comparison. The same interpolation type (Catmull-Rom) and parameters were used. It was chosen an inverted colormap for easier visualisation of potential artifacts. On the **right**, it is shown the image subtraction between both reconstructions.

The resultant reconstructions exhibit distinct characteristics, although they maintain a degree of similarity. This expected resemblance is due to the shared mathematical foundation underpinning both implementations. However, the differences shown might be attributed to subtle variations in numerical computations and precision handling, inherently different in both programming languages. However, there is a clear presence of macro-pixel patterning in the NanoJ reconstruction, which is absent in NanoPyx. The direct subtraction of one reconstruction from the other underscores the variations between them. The highlighted microtubule structure, which arises from a pixel shift between the two, is the most notable difference. Importantly, the lack of substantial differences further reinforces confidence in the method's precise implementation

The following analysis delves into the implementations of each step involved in eSRRF: Catmull Rom interpolations, Roberts cross gradient calculation and RGC map calculation. Each of them was implemented as a Liquid Engine task, using the NanoPyx metaprogramming system for the generation of multiple code variations.

### Interpolations

The selection and application of the interpolation technique influence the reconstruction outcome, potentially giving rise to artifacts. In the context of the NanoJ-eSRRF implementation, the user is given the choice of either using the Catmull-Rom interpolation (used in the previous SRRF), or the FHT interpolation, for image upsampling. This choice was prompted by the observed macro-pixel artifacts arising from the use of Catmull-Rom interpolation (visible in Figure A.7). The reason why the FHT is not universally chosen is because it introduces a significant overhead in execution speed, as it is a more computationally complex task. In NanoPyx, in initial experiments, the Catmull-Rom interpolation was used, and surprisingly, the macro-pixel patterning was not visible in the final reconstruction (Figure A.7). This comparison led to the determination that the Catmull-Rom interpolation should be uniformly adopted for all interpolation steps within the NanoPyx implementation.

The Catmull-Rom interpolation was implemented in the Liquid Engine, where a lower-level C function was created to interpolate each pixel of the image. It then uses for in range loops to iterate over the image pixels and apply the interpolation function to each pixel. The CPU-based implementations were created by using the tag2tag meta-programming tool, as described in section 2.5. The implementation initially written was with a single core of the CPU, and then the tag2tag tool was used to generate the parallelised code, in which automatically substituted the location of the "range" with "prange", allowing multi-threading. The following snippet illustrates part of a run function of the Catmull-Rom interpolation in the Liquid Engine, with the parallelisation applied in the rows of the image.

```
for f in range(nFrames):
    for j in prange(height_magnified, schedule=schedule_type):
        col = j / magnification
        for i in range(width_magnified):
            row = i / magnification
            image_out[f,i,j] = _c_interpolate(image_in, row, col)
```

The tag2tag tool allowed the propagation of 5 CPU-based implementations (for a single-core, and multi-core with the four different schedulers `static`, `dynamic`, `guided` and `auto`). In addition, a GPU-based implementation was created, where a pyOpenCL kernel was built.

### Gradient Calculation

Another crucial step in eSRRF is to calculate the intensity gradients of the input image. This was done by using the Roberts Cross operator. The Roberts Cross operator is a 2x2 convolution diagonal kernel (in both diagonal directions), and it was implemented as a pure C function and applied to all the pixels in each image frame. The resulting images were rotated by 45º in order to obtain the horizontal (`Gx`) and vertical (`Gy`) intensity gradients, which are the output of this step. The gradient calculation task was implemented in the Liquid Engine, where a similar approach was used as in the Catmull-Rom interpolation. The CPU-based implementations were created by using the tag2tag tool, and the parallelisation was done in the calculation of individual frames, as follows.

```
for n in prange(nFrames, schedule=schedule_type):
    _c_gradient_roberts_cross(image)
```

Moreover, similarly to the interpolations, 5 CPU-based implementations generated and 1 GPU-based implementation was created.

### Radial Gradient Convergence Map Calculation

The last step of the eSRRF spatial analysis is the calculation of the RGC maps, based on the interpolated gradients stacks. A C function was built to calculate the radial gradient convergence per subpixel. It outputs the RGC value, calculated as described in 1.1.5. Then, each calculated RGC value is assigned to the corresponding pixel in the output image (the RGC map).

This step was also implemented into the Liquid Engine, where the parallelisation for the CPU-based implementations is done in the rows of the image, as follows.

```
for f in range(nFrames):
    for j in prange(height_magnified, schedule=schedule_type):
        for i in range(width_magnified):
            RGC_map[f, j, i] = _c_calculate_rgc(interp_grad_x, interp_grad_y)
```

Following the same pattern as the preceding stages, a set of 5 CPU-based implementations and 1 GPU-based implementation were generated.

The aforementioned steps constitute the spatial analysis undertaken by the eSRRF algorithm. These steps have been seamlessly integrated into a workflow, as previously outlined in A.2.2. However, the temporal analysis between the reconstructed images has yet to be addressed. In the NanoJ version, this involves three potential correlations: the mean, variance, and $2^{nd}$ order autocorrelation function.

It was determined that these correlations would be implemented separately from the spatial analysis. Presently, these temporal analyses are not incorporated within the Liquid Engine framework. This decision was influenced by the use of the numpy Python library for these computations, employing functions such as numpy.mean and numpy.var, along with a straightforward Python function for calculating the 2nd order autocorrelation function. Notably, rigorous testing indicated that even for extensive datasets, such as the one featured in Figure 2.9, these operations did not introduce any notable overhead. This efficiency can be attributed to the optimization features inherent to the numpy library.

### Integration on a Liquid Engine Workflow

In NanoPyx, eSRRF was implemented as a single task of a Liquid Engine workflow, meaning that once eSRRF is ran, the same run type is used for all its individual steps.

```
eSRRF_WF = Workflow(eSRRF(image, magnification, sensitivity, do_intensity_weighting))
```

The choice of consolidating eSRRF as a single task within the Liquid Engine stems from a practical consideration related to GPU use. In scenarios where the GPU is identified as the fastest option, a series of performance benchmarks revealed that the cumulative overhead associated with individually executing each step of the eSRRF method on the GPU is substantial. This is attributed to the repeated data transfers between the CPU and GPU during these steps. Surprisingly, the time spent on moving data back and forth between the CPU and GPU during each step becomes a more significant performance bottleneck than the actual computational workload of those steps. Consequently, an alternative approach was devised: by maintaining eSRRF as a unified task, the impact of data transfer overhead can be mitigated. In this configuration, if the GPU is determined as the optimal choice, the data is initially transferred to the GPU, all eSRRF steps are executed on the GPU, and the results are eventually transferred back to the CPU. In order to illustrate this, a benchmarking analysis was conducted that compared two scenarios: one where individual steps of eSRRF were executed as tasks within the Liquid Engine, and another where the entire eSRRF process was treated as a single task. The results revealed a significant speedup of 11 times, reducing the processing time from 31 seconds to 1.9 seconds (in a 500x128x128 image).

The following listing shows the structure of the eSRRF task, containing all the previously described spatial analysis steps.

```
run_type = X
crsm = CatmullRomInterpolation()
robx = GradientRobertsCross()
rgc = RadialGradientConvergence()

magnified_image = crsm.run(image, magnification, run_type)
gradient_x, gradient_y = robx.run(image, run_type)
gradient_x_interp = crsm.run(gradient_x, 2* magnification, run_type)
gradient_y_interp = crsm.run(gradient_y, 2* magnification, run_type)
rgc_map = rgc.run(gradient_x_interp, gradient_y_interp, magnified_image, runtype)
```

Extensive testing of all implemented steps was conducted using both simulated and real data, as illustrated in Figures 2.9, A.6, and A.7. After being fully integrated as a Liquid Engine workflow, the NanoPyx eSRRF method was employed not only to validate the decision-making process of the Liquid Engine agent (see Supplementary Figure 2.8) but also to compare its computational efficiency with the NanoJ version. With the same parameters and interpolation techniques, the results revealed a significant 2.5 times speedup on the dataset employed within the demonstrated NanoPyx framework (Figure 2.9).

## A.3 Monte Carlo Particle Simulation

Within the NanoPyx platform, an algorithm was developed to simulate a two-dimensional particle field based on a user-defined input PDF, as outlined in Algorithm A.1.

It was designed to accept any two-dimensional image as a potential ground truth and offers adjustable default parameters, including the maximum number of particles for simulation, minimum distance between particles, maximum number of tries, and the mean distance threshold between particles.

The devised algorithm is rooted in the principles of the Monte Carlo method. It randomly generates particles and places them in locations with a probability that is proportional to the intensity of the input PDF at that specific location. Once a list of particle positions is generated, various stopping criteria are evaluated. These criteria involve conditions like reaching the maximum number of particles (max_particles), achieving a mean distance between particles below the set threshold (mean_distance_threshold), or exhausting the maximum of tries for the simulation (nb_tries). If any of these conditions are satisfied, the algorithm terminates and returns the list of particle positions. Otherwise, it proceeds to the subsequent iteration, introducing more particles placed randomly and proportionally to the input PDF.

**Algorithm A.1:** Simulate 2D Particle Field

```
Input  : Ground-truth image (probability density function)
Output: Particle positions (x, y)
while true do
    particles_not_set → list of indexes of non set particles
    for p in particles_not_set do
        get_particle_candidate → returns 0 or 1 whether a particle was placed, with a
          probability proportional to the sampled PDF
    particles_set → list of indexes of set particles
    for p in particles_set do
        get_closest_distance(x[p], y[p], x, y) → calculates closest distance between set
          particles
        if closest_distance < _min_distance then
            Kick particles x[p] and y[p] out (they were found too close)
        else
            n_particles += 1 → calculate number of set particles
    if n_particles = previous_n_particles then
        nb_tries += 1 → increment number of tries if no new particles were set
    else
        nb_tries = 0 → reset number of tries if new particles were set
    if n_particles = max_particles or tries = max_tries then
        break
    if mean_distance_threshold > 0 and n_particles > min_particles and mean_closest_distance
      < mean_distance_threshold then
        break
x = x[particles_set]
y = y[particles_set]
return np.array([x, y]) → array of final set particle positions
```

Upon the application of the algorithm to the input ground truth, a list of particle positions is generated as output.

## A.4 Competitive Labelling Molecular Counts

To normalise the number of detections in both the microtubule and noise regions, the areas of these regions were measured and considered in the calculation of the SNR. The noise region had a fixed area of 1 $\mu$m$^2$, whereas the length of the microtubule region was adjusted to be 1 $\mu$m after being divided by its measured length. The width of the microtubule region was determined to be 187 nm, calculated from 8 subpixels, each with a size of 23.4 nm (corresponding to 117 nm pixel divided by the used magnification of 5). Therefore, the area of the microtubule region was (measured length) $\mu$m x 0.187 $\mu$m. This was considered and accounted for in the calculation of the SNR, ensuring a fair and equitable comparison between different labelling conditions. For this study, it is assumed the SNR is calculated as follows.

$$\text{SNR} = \frac{\text{Nb detections microtubule / area microtubule}}{\text{Nb detections noise / area noise}} = \frac{\text{Nb detections microtubule / (length x 0.187)}}{\text{Nb detections in noise / 1}}$$

The results of the number of detections along microtubules, including their measured lengths, and the number of detections in a squared micrometer noise area for each paired competition ratio are depicted in Table A.1 and Table A.2.

For each paired competition ratio, the analysis included a total of 9 measurements of the number of detections per micrometer. This was achieved by analysing 3 cells for each ratio, and within each cell, 3 microtubules were considered. However, when calculating the SNR, only 3 of these measurements were taken into account. Specifically, the measurements were chosen from areas in the sample with lower local microtubule density, meaning regions where fewer microtubules were present nearby. This was done to ensure that the number of detections in the microtubule region was not influenced by the presence of neighbouring microtubules, which could potentially introduce bias in the results.

**Table A.1:** Obtained results for the labelled/ unlabelled ratios of 0.1, 0.2, 0.3, 0.4 and 0.5. For each ratio, 3 cells were considered, and in turn, for each cell 3 microtubule (MT) and noise areas were selected. The number of detections per microtubule and noise are shown, including the measured lengths and areas for normalisation. The calculated mean SNR and standard deviation are presented in the right column.

| Ratio | Cell | MT detections / length | Noise detections | Mean SNR ± StdDev |
|---|---|---|---|---|
| 0.1 | 1 | 991 / 2.109 | 683 | 2.452 ± 0.148 |
| | | 1502 / 1.643 | 864 | |
| | | 1321 / 1.791 | 1123 | |
| | 2 | 835 / 1.085 | 1762 | |
| | | 884 / 1.228 | 793 | |
| | | 489 / 1.023 | 961 | |
| | 3 | 898 / 1.269 | 1331 | |
| | | 1025 / 0.985 | 1185 | |
| | | 333 / 0.927 | 815 | |
| 0.2 | 1 | 1867 / 1.319 | 707 | 6.323 ± 0.069 |
| | | 902 / 1.084 | 857 | |
| | | 1401 / 1.320 | 2331 | |
| | 2 | 2373 / 1.370 | 1487 | |
| | | 1788 / 1.432 | 1054 | |
| | | 1533 / 1.019 | 1258 | |
| | 3 | 1079 / 1.001 | 806 | |
| | | 768 / 1.143 | 197 | |
| | | 1399 / 1.162 | 323 | |
| 0.3 | 1 | 1910 / 1.511 | 1151 | 8.068 ± 0.322 |
| | | 2308 / 1.402 | 1039 | |
| | | 3434 / 1.288 | 1855 | |
| | 2 | 2383 / 1.097 | 1446 | |
| | | 1449 / 1.049 | 438 | |
| | | 1172 / 1.557 | 612 | |
| | 3 | 1681 / 1.471 | 1519 | |
| | | 2409 / 1.130 | 1740 | |
| | | 1501 / 1.268 | 1991 | |
| 0.4 | 1 | 2716 / 1.662 | 921 | 10.143 ± 0.663 |
| | | 2896 / 1.458 | 693 | |
| | | 2147 / 1.456 | 597 | |
| | 2 | 2721 / 1.171 | 1258 | |
| | | 2869 / 1.177 | 2002 | |
| | | 3806 / 1.254 | 1469 | |
| | 3 | 3040 / 1.276 | 1230 | |
| | | 2620 / 1.389 | 1870 | |
| | | 2940 / 1.564 | 1243 | |
| 0.5 | 1 | 8263 /1.795 | 1390 | 11.380 ± 0.506 |
| | | 5785 / 1.574 | 1680 | |
| | | 7439 / 1.696 | 1597 | |
| | 2 | 7635 / 1.345 | 2580 | |
| | | 6821 / 1.550 | 4810 | |
| | | 12218 / 1.630 | 4304 | |
| | 3 | 4283 / 1.919 | 712 | |
| | | 2279 / 1.197 | 955 | |
| | | 1268 / 1.429 | 847 | |

**Table A.2:** Obtained results for the labelled/ unlabelled ratios of 0.6, 0.7, 0.8, 0.9 and 1.0. For each ratio, 3 cells were considered, and in turn, for each cell 3 microtubule (MT) and noise areas were selected. The number of detections per microtubule and noise are shown, including the measured lengths and areas for normalisation.

| Ratio | Cell | MT detections / length | Noise detections | Mean SNR +- StdDev |
|---|---|---|---|---|
| 0.6 | 1 | 4707 /1.843 | 1097 | 11.964 ± 0.385 |
| | | 2367 / 1.341 | 580 | |
| | | 2735 /1.434 | 886 | |
| | 2 | 2840 / 1.372 | 929 | |
| | | 2459 / 1.435 | 850 | |
| | | 3590 / 1.568 | 990 | |
| | 3 | 3469 / 1.456 | 1021 | |
| | | 2201 / 1.134 | 850 | |
| | | 2876 / 1.238 | 922 | |
| 0.7 | 1 | 8324 / 2.115 | 1703 | 12.651 ± 0.213 |
| | | 4202 / 1.759 | 1005 | |
| | | 5690 / 1.944 | 1238 | |
| | 2 | 1743 / 1.592 | 455 | |
| | | 1934 / 1.485 | 439 | |
| | | 2948 / 1.658 | 560 | |
| | 3 | 3280 / 1.249 | 850 | |
| | | 4570 / 2.008 | 984 | |
| | | 3847 / 1.547 | 842 | |
| 0.8 | 1 | 4925 / 1.489 | 1374 | 13.070 ± 0.142 |
| | | 5910 / 1.746 | 1370 | |
| | | 6023 / 1.804 | 1451 | |
| | 2 | 1917 / 1.664 | 470 | |
| | | 2209 / 1.703 | 495 | |
| | | 1993 / 1.605 | 341 | |
| | 3 | 4509 / 1.869 | 982 | |
| | | 4203 / 1.923 | 1103 | |
| | | 2980 / 1.170 | 879 | |
| 0.9 | 1 | 3271 / 1.476 | 728 | 13.154 ± 0.476 |
| | | 2226 / 1.691 | 744 | |
| | | 2759 / 1.738 | 618 | |
| | 2 | 3426 / 1.433 | 1017 | |
| | | 3149 / 1.464 | 1126 | |
| | | 4355 / 1.493 | 478 | |
| | 3 | 6298 / 1.826 | 1404 | |
| | | 4376 / 1.317 | 1491 | |
| | | 3376 / 1.281 | 1810 | |
| 1.0 | 1 | 6367 / 1.672 | 1023 | 12.855 ± 0.833 |
| | | 9545 / 1.997 | 1918 | |
| | | 7263 / 1.647 | 1032 | |
| | 2 | 7205 / 1.519 | 3571 | |
| | | 7502 / 1.811 | 3830 | |
| | | 10301 / 1.604 | 1509 | |
| | 3 | 4212 / 1.182 | 1407 | |
| | | 6263 / 1.394 | 2715 | |
| | | 9572 / 1.516 | 2891 | |