Physics-Inspired Single-Particle Tracking Accelerated with Parallelism

Lance W.Q. Xu (徐伟青)[1,2] and Steve Pressé[1,2,3,*]

[1]Center for Biological Physics, Arizona State University, Tempe, AZ, USA
[2]Department of Physics, Arizona State University, Tempe, AZ, USA
[3]School of Molecular Sciences, Arizona State University, Tempe, AZ, USA
[*]Corresponding author (email: spresse@asu.edu)

Abstract

Data modeling tools face trade-offs between accuracy, computational efficiency, data efficiency, and model flexibility. Physics-inspired, rigorous likelihood-based approaches, while offering high accuracy and data efficiency, remain limited in practice due to high computational cost, particularly when applied to larger-scale problems. This general limitation is further compounded by reliance on traditionally single-threaded iterative sampling or optimization procedures, which are difficult to scale. Although prior efforts have attempted to parallelize expensive likelihood-based approaches by partitioning data or running multiple sampling replicas in parallel, such strategies fail for algorithms requiring efficient communication between processes. Here, we introduce a fundamentally different strategy: we exploit the parallelism inherent in both likelihood evaluation and posterior sampling, operating on a single shared dataset. Our framework supports frequent yet lightweight inter-thread and inter-processor communication, making it well-suited for modern parallel architectures. Using diffraction-limited single-particle fluorescence tracking as a case study, this approach achieves up to a 50-fold speedup on a single mid-range GPU compared to conventional single-threaded CPU implementations, demonstrating a scalable and efficient solution for high-performance likelihood-based inference.

1 Introduction

Data modeling tools face key trade-offs between accuracy, computational efficiency, data efficiency, and model flexibility; see Fig. 1a. Among these, physics-inspired likelihood-based approaches represent an essential class due to their foundation in physical principles and mathematical rigor. These methods often yield highly accurate and data-efficient results by

directly modeling the generative processes behind observed data[1–17]. However, likelihood-based methods present key limitations. First, constructing a precise likelihood function requires a detailed understanding of the system's physics, which can restrict the model's adaptability to different experimental modalities. Second, these methods tend to be computationally intensive, mainly when applied to setups where evaluating the likelihood is computationally demanding and analytically intractable.

To address this computational burden, many methods rely on divide-and-conquer strategies, splitting large datasets into smaller subsets that can be analyzed independently[18–25]. While effective for some tasks, this strategy breaks down when applied to data with strong spatiotemporal correlations, where dividing the data risks discarding essential contextual information. This limitation is particularly relevant across a wide range of scientific disciplines. For instance, particle image velocimetry analyzes flow fields based on tracer particle motion in image sequences[26], and behavioral studies in neuroscience or ethology rely on continuous video analysis of animal or human subjects[27,28]. In such contexts, accelerating inference without sacrificing information contained in the correlation structure of the data is a key challenge.

Widefield single-particle tracking (SPT) is another representative example of these challenges. Mathematically, a particle's trajectory is represented as a sequence of spatial positions over time. The objective is to infer the full probability distribution over the number of particle tracks and their associated trajectories, conditioned on the recorded image frames and the physical principles governing the system. This, distribution, termed posterior, is shaped by two key components: the emission model (serving as the likelihood), which quantifies the probability of observing the recorded frames given a set of particle tracks, and the motion model (serving as the prior), which describes the plausibility of the tracks themselves independent of the data.

Accurately evaluating the emission model requires calculating the expected photon emissions from all particles across every pixel and frame, owing to the spreading of light caused by diffraction and the detector's physical properties. As the number of particles, pixels, and frames increase, this task becomes computationally intensive. In addition, posterior inference typically relies on iterative sampling techniques—most notably, Markov chain Monte Carlo (MCMC)[29]—which, though statistically rigorous, demand repeated evaluations of costly likelihood functions. In our previous work[30], we demonstrated a physics-inspired SPT framework, BNP-Track, that yields high tracking accuracy and strong data efficiency, particularly under challenging imaging conditions. However, it comes at a significant computational cost: generating sufficient posterior samples often takes several hours on a standard desktop CPU. This contrasts with conventional tools such as TrackMate[31], which complete inference in seconds. Such computational demands limit the practicality of physics-inspired methods in high-throughput applications, particularly those involving larger biological and imaging datasets.

Approximations are often invoked to help mitigate computational expense. For instance, in the context of particle tracking, a common simplification involves focusing solely on identifying the most probable set of trajectories, often without quantifying uncertainty[31–33].

Another widely adopted approximation relies on greedy algorithm structures decomposing tracking into sequential steps of localization and linking[31–33]. In the localization step, object positions are detected within each frame independently of the other, extracting spatial coordinates and timestamps. The linking step then connects these localized positions across frames to reconstruct full particle trajectories. By reducing a high-dimensional inference problem into a series of lower-dimensional ones, this modular design enables fast computation, often on the order of seconds per dataset. Moreover, it offers considerable flexibility, allowing researchers to tailor the method by mixing and matching localization and linking algorithms to suit various experimental conditions.

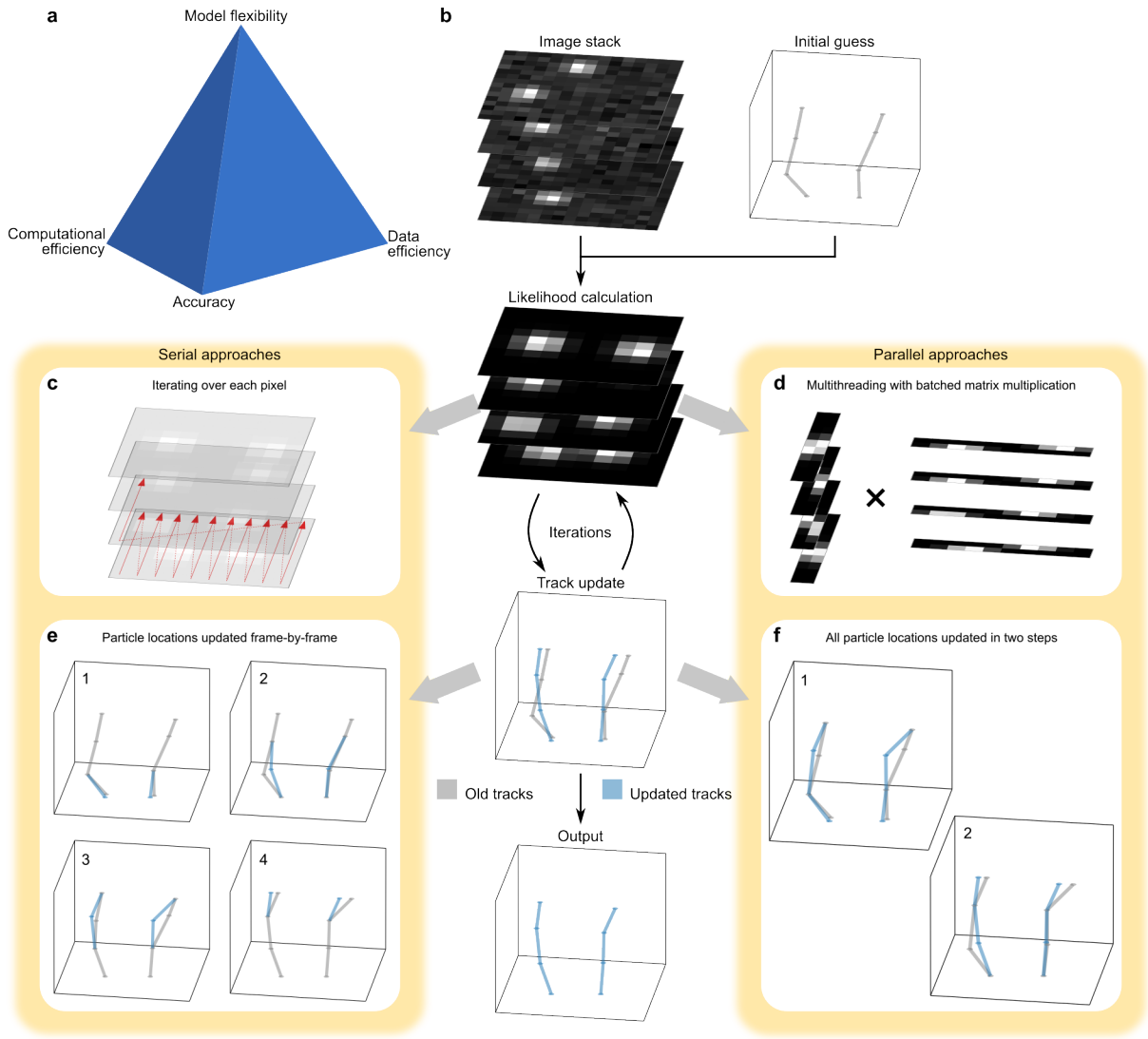

Figure 1: **(a)**, Particle tracking methods often face trade-offs among four key aspects: tracking accuracy, computational efficiency, data efficiency, and model flexibility. **(b)**, In the Bayesian paradigm, tracking begins with a recorded image stack and initial track estimates, followed by iterative updates of particle tracks. These updates rely on evaluating the likelihood of each proposed set of tracks. **(c)**, Likelihood calculations can be performed serially by evaluating each pixel individually. **(d)**, Using batched matrix multiplication allows for parallelized likelihood computation, which can more efficiently leverage thousands of threads available in modern GPUs. **(e)**, A naive approach to updating tracks involves frame-by-frame serial updates. **(f)**, With improved algorithmic design, particle positions at even and odd frames can be updated in just two steps, regardless of how many frames there are in total.

While these approximations offer substantial gains in computational speed and model

flexibility, they often come at the cost of reduced tracking accuracy[30]. High-precision localization requires a sufficient number of detected photons per frame[34,35], a condition difficult to satisfy in live-cell imaging. Increasing illumination intensity to boost photon counts risks photodamage[36], while extending exposure times introduces motion blur, particularly problematic for fast-diffusing biomolecules. As a workaround, datasets are frequently filtered to include only the brightest signals, excluding weaker ones that may still be biologically relevant. Such filtering introduces selection bias and reduces data efficiency by discarding valuable information embedded in lower-intensity signals.

Beyond approximation-based strategies, deep learning approaches for bioimaging[37–40] as well as particle tracking offer another computationally attractive option[41–45]. Leveraging deep neural networks, these methods offer high model flexibility and, once trained, are computationally efficient. However, their performance is often sensitive to variations in signal-to-noise ratio (SNR)[45], and they typically require large, carefully curated training datasets—either simulated or experimentally annotated. As a result, their tracking accuracy and data efficiency can be limited, particularly in challenging or previously unseen imaging conditions[30,32].

In addition to approximation-based strategies, researchers have explored various parallelization techniques to accelerate likelihood-based inference methods[18–23,25,46–50]. Broadly, these approaches fall into two categories: methods that partition the dataset into independent subsets and analyze each in parallel before aggregating the results[18–23,25], and methods that replicate the full dataset across multiple threads or processes and combine outputs from independent runs[25,46–50].

While both strategies can accelerate convergence and are often described as embarrassingly parallel, they fall short in settings where frequent inter-process communication is required. This limitation is particularly acute in particle tracking problems, where per-pixel likelihoods must be aggregated to compute frame-level likelihoods, which in turn must be combined to evaluate the full system likelihood. Additionally, strong temporal correlations between particle positions across frames introduce statistical dependencies demanding frequent synchronization and global coordination. As a result, communication overhead becomes the dominant bottleneck, limiting the efficiency gains of parallelization for single-chain inference. Furthermore, combining results from parallel runs typically requires *post hoc* approximations, such as weighted averaging of subposterior samples (posteriors of sub-datasets)[20] or Gaussian process fit on estimated density values[21], which may compromise the statistical rigor that fully Bayesian methods aim to preserve.

To address these challenges while preserving the full mathematical rigor and tracking accuracy, we present a parallel SPT framework, BNP-Track 2.0. This implementation capitalizes on the intrinsic parallelism of both likelihood evaluation and posterior sampling by leveraging modern computing architectures, including vectorized operations, multi-threading, and GPU acceleration. By supporting frequent yet lightweight communication across threads (see Fig. 1b), the parallel framework achieves a 50-fold speedup on a mid-range GPU (Nvidia GTX 1060, 6 GB) relative to the original serial framework executed on a similarly priced CPU (Intel i7-7700K). Importantly, this acceleration comes with no data efficiency or inference

quality loss, making our framework a practical and scalable solution for high-throughput particle tracking in complex biological environments.

2 Results

We begin by demonstrating that the new parallel physics-inspired SPT framework retains the same high level of tracking accuracy as its predecessor, particularly in low SNR conditions. At the same time, it delivers a substantial computational boost to rigorous likelihood-based approaches—achieving nearly a 50-fold speedup when run on a mid-range Nvidia GTX 1060 (6 GB) GPU compared to the original serial implementation on an Intel i7-7700K CPU. We then perform a head-to-head comparison of the two frameworks across various parameter regimes, highlighting how their computational performance scales under different conditions. Finally, we show that when the number of emitting particles is fixed (*i.e.*, in a parametric setting), the parallel framework achieves an additional 20-fold speedup, bringing the total acceleration to nearly 200-fold on a single GTX 1060 GPU.

2.1 Tracking with low SNR data

In the first robustness test, we verify that the parallel physics-inspired SPT framework preserves its predecessor's tracking accuracy even for low SNR. We demonstrate this by simulating a series of image stacks sharing the same parameters, including particle tracks, field of view (FOV), pixel size, emission wavelength, numerical aperture, refractive index, and emission rate, and only changing the background level. Each image stack contains ten 128×128 frames and ten diffusing particles. For illustrative purposes, Fig. 2a shows the resulting time-averaged frames; the noise level of the rightmost dataset is 20 times as high as that of the leftmost dataset.

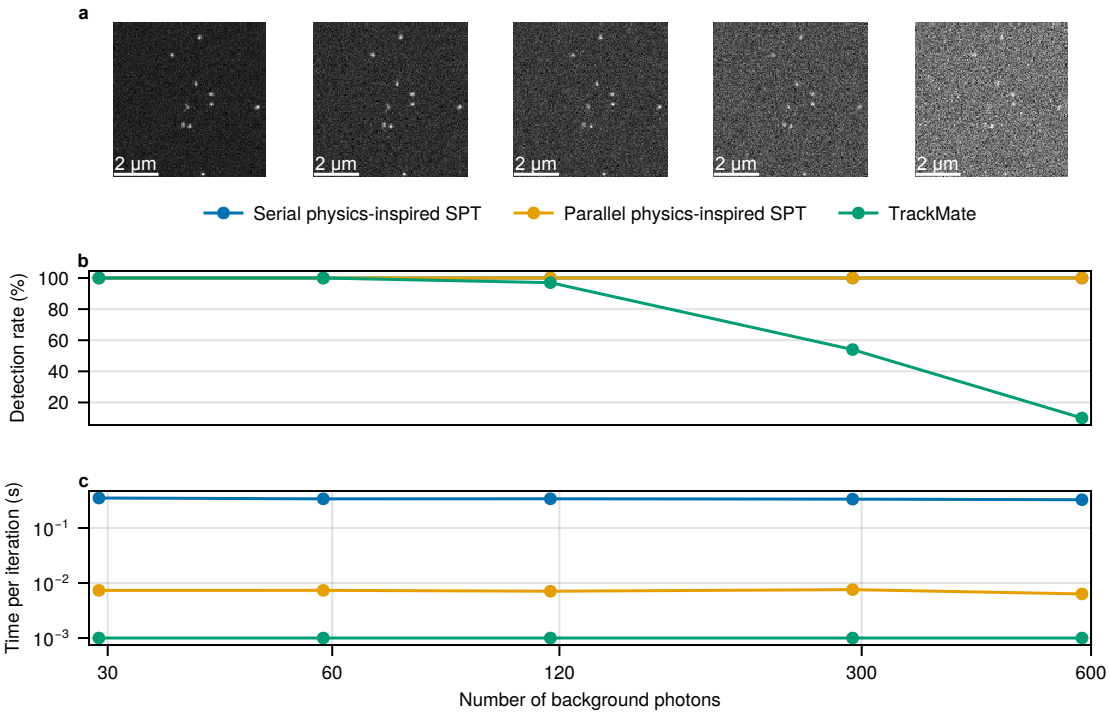

Figure 2: Tracking accuracy and computational efficiency comparisons between the serial physics-inspired SPT, the parallel physics-inspired SPT, and TrackMate[31] under varying background noise levels for synthetic data. **(a)**, Time-averaged frames corresponding to different background noise levels in terms of the number of background photons per pixel per frame, with contrast adjusted for visualization. Each emitting particle contributes roughly 80 photons to the pixel concentric with the particle's point spread function (PSF), consistent with Cy3 dye brightness[30,51]. **(b)**, Detection ratio for each method, representing tracking accuracy based on the proportion of true particle detections without spurious results. Note that the detection ratios of the physics-inspired frames (blue and orange) are the same at all noise levels. **(c)**, Per-iteration runtime for each method, representing computational efficiency. The simulated data assumes a numerical aperture of 1.45, a refractive index of 1.515, an emission wavelength of 665 nm, a frame exposure time of 33 ms, and a pixel size of 133 nm, using an EMCCD camera with an offset and EM gain of approximately 100. Each dataset consists of ten 128×128 frames with particles diffusing at 0.1 $\mu m^2/s$. All benchmarks were conducted on the same desktop computer with Intel i7-7700K CPU and Nvidia GTX 1060 6 GB GPU, see Computational efficiency measurement.

As discussed earlier, many parallel inference methods introduce approximations that can compromise accuracy. It is therefore essential to demonstrate that the proposed parallel physics-inspired SPT framework preserves the accuracy of its serial predecessor. To this end, we compare the tracking performance of the parallel framework, the original serial framework,

and TrackMate[31], a widely used tool in the literature[52–56]. Specifically, we evaluate each method's accuracy by tuning parameters to ensure that no particle detections occur more than one diffraction limit away from any ground truth position (termed spurious detections). Once this criterion is satisfied, the detection ratio is calculated as the number of correct detections divided by the total number of ground truth positions. As shown in Fig. 2b, both the original and the parallel physics-inspired SPT frameworks reliably detect all particles across all frames. At the same time, TrackMate's performance degrades significantly as background noise increases.

Having evaluated tracking accuracy using the detection ratio, we now assess computational efficiency. Unlike accuracy metrics, comparing computational efficiency across methods is more nuanced due to inherent differences in what each algorithm infers from the data. For instance, the physics-inspired frameworks estimate particle trajectories and quantify the uncertainty associated with each track—an important capability absent in tools like TrackMate.

Moreover, computational efficiency alone does not fully reflect the practical cost of analysis, particularly for methods requiring extensive manual parameter tuning. TrackMate, for example, asks users to specify parameters such as localization quality thresholds and maximum linking distances. Identifying suitable values often involves a time-consuming, trial-and-error process, the duration of which varies across users and datasets and is challenging to quantify systematically. In contrast, physics-inspired methods generally involve fewer tunable hyperparameters[5,29,30,57,58], significantly reducing the burden on the user. These methods are typically designed to be fully automated once the input is provided[5,17,57,59–63], meaning that all time costs are confined to computation itself, with no manual intervention required during execution. As a result, these approaches improve computational efficiency and minimize user effort and variability.

Despite these caveats, comparing per-iteration runtime remains informative, especially in highlighting algorithmic and implementation differences. This comparison is straightforward for physics-inspired SPT frameworks with and without parallelism: both rely on MCMC sampling and follow the same inference logic, requiring a comparable number of iterations to achieve equivalent tracking accuracy. For TrackMate, we define per-iteration runtime as the time taken for one round of localization and linking given a fixed set of parameters—excluding time spent on user tuning—to allow for a fairer comparison of algorithmic performance.

In 2c, we present the per-iteration runtime for each method. Across all signal-to-noise ratio (SNR) regimes, the three methods show similar per-iteration times—an expected result, as SNR primarily affects the number of iterations required for convergence rather than the time per iteration.

TrackMate exhibits the shortest per-iteration time, typically under 10 ms (Intel i7-7700K, 4 cores, 8 threads, up to 4.5 GHz). However, as discussed earlier, this timing excludes the manual effort required for parameter tuning. In contrast, the physics-inspired framework with parallelism, which is algorithmically comparable to its predecessor, achieves over a nearly 50-fold improvement in computational speed on a Nividia 1060 6 GB GPU (see Computational efficiency measurement), underscoring the benefits of the optimizations imple-

mented in the new version.

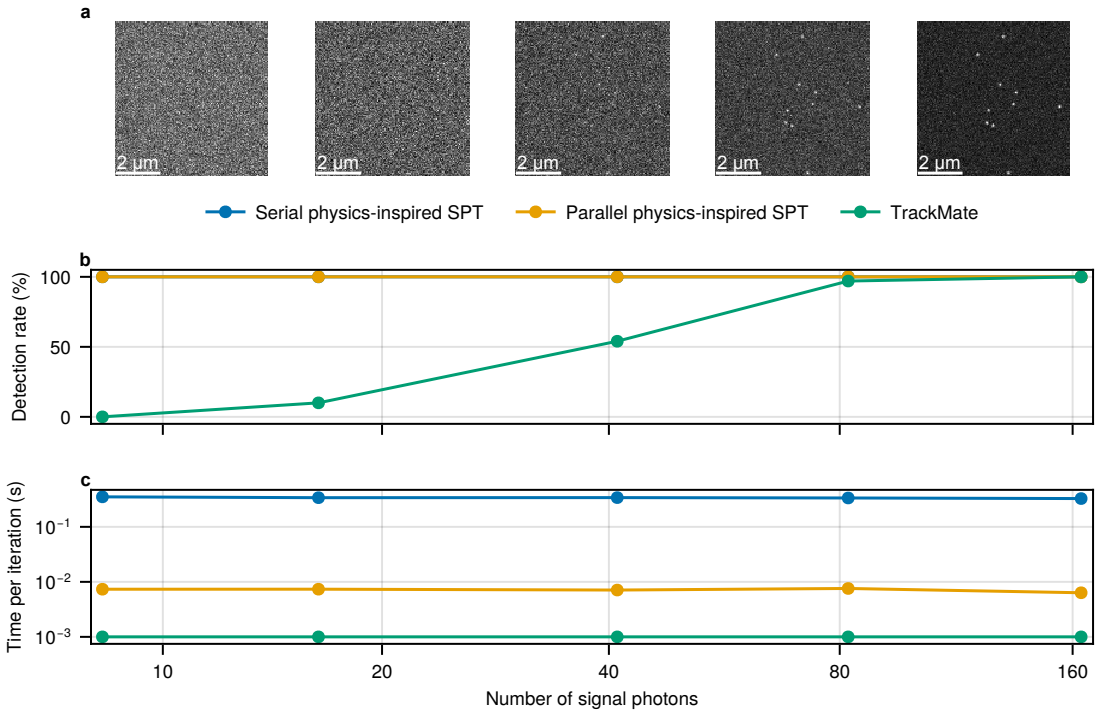

Figure 3: Tracking accuracy and computational efficiency comparisons between the serial physics-inspired SPT, the parallel physics-inspired SPT, and TrackMate[31] under varying particle emission rates using synthetic data. **(a)**, Time-averaged frames corresponding to different emission rates in terms of the number of photons received at the pixel concentric with the center PSF. The number of 80 photons is consistent with Cy3 dye brightness[30,51]. The amount of background is fixed at around 60 photons per pixel per frame. **(b)**, Detection ratio for each method, representing tracking accuracy based on the proportion of true particle detections without spurious results. Note that the detection ratios of the physics-inspired frames (blue and orange) are the same at all emission rates. **(c)**, Per-iteration runtime for each method, representing computational efficiency. The simulated data assumes a numerical aperture of 1.45, a refractive index of 1.515, an emission wavelength of 665 nm, a frame exposure time of 33 ms, and a pixel size of 133 nm, using an EMCCD camera with an offset and EM gain of approximately 100. Each dataset consists of ten 128×128 frames with particles diffusing at 0.1 $\mu m^2/s$. All benchmarks were conducted on the same desktop computer with Intel i7-7700K CPU and Nvidia GTX 1060 6 GB GPU, see Computational efficiency measurement.

In addition to the tests shown in Fig. 2, we conducted a complementary experiment in which the brightness of each emitting particle varied across datasets. The results, shown in Fig. 3, are consistent with those in Fig. 2, further confirming that the parallel physics-inspired

framework maintains its performance while being nearly 50 times more computationally efficient.

2.2 Computational efficiency scaling

In the previous section, we demonstrated the computational efficiency gains achieved by the parallel physics-inspired SPT framework. Here, we present a more detailed comparison with its serial counterpart, evaluating how performance scales with frame size, number of frames, and number of emitting particles. In addition to benchmarking the parallel framework on a GPU, we also report its performance on a CPU for a comprehensive assessment across hardware platforms.

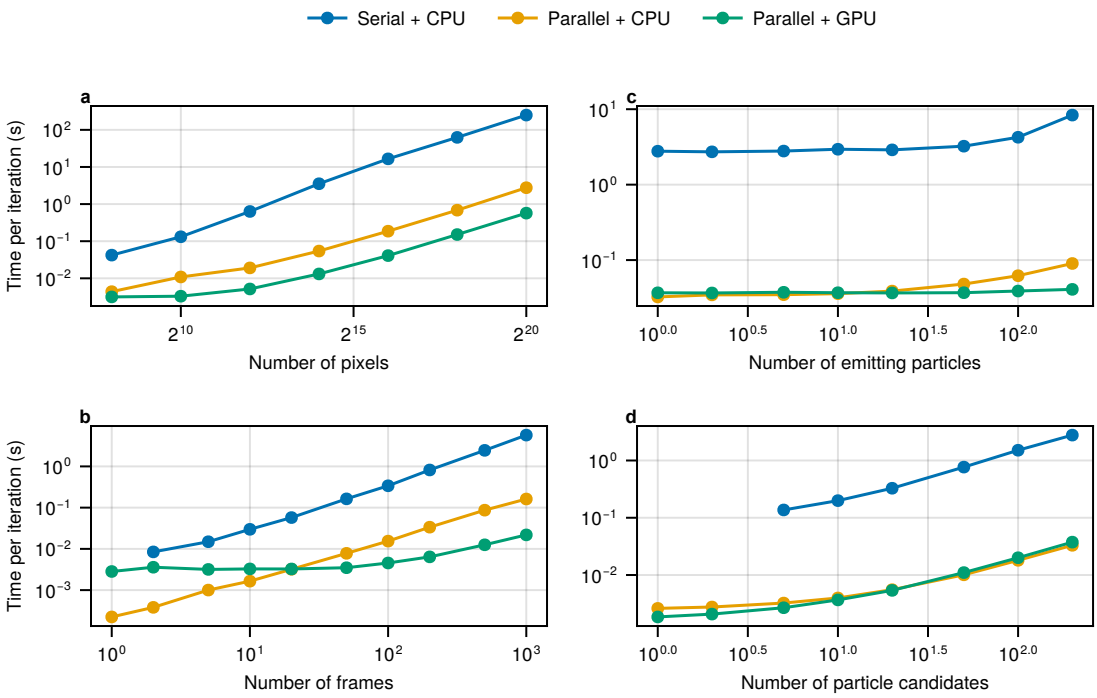

Figure 4: Scalability of computational efficiency for the serial physics-inspired SPT framework on CPU (blue), the parallel framework on CPU (orange), and the parallel framework on GPU (green), across various parameters. **(a)**, Frame size: per-iteration runtime measured as frame dimensions increase from 16×16 to 1024×1024. **(b)**, Number of frames: per-iteration runtime as the number of analyzed frames increases from 1 to 1000. **(c)**, Number of emitting particles: per-iteration runtime as the number of emitting particles increases from 1 to 200 while keeping the frame size and count fixed. **(d)**, Total number of particle candidates: per-iteration runtime when increasing the number of candidates (real and inactive) from 1 to 200 while keeping only one particle active. The simulated data assumes a numerical aperture of 1.45, a refractive index of 1.515, an emission wavelength of 665 nm, a frame exposure time of 33 ms, and a pixel size of 133 nm, using an EMCCD camera with an offset and EM gain of approximately 100. All benchmarks were conducted on the same desktop computer with Intel i7-7700K CPU and Nvidia GTX 1060 6 GB GPU, see Computational efficiency measurement.

In Fig. 4a, we simulate 100 frames, each containing two emitting particles, with frame sizes ranging from 16×16 to 1024×1024 pixels. The parallel physics-inspired SPT framework maintains a tenfold speed advantage on the CPU over its serial counterpart, demonstrating better scaling as frame size increases. This improvement stems from its optimized algorithmic structure, detailed in Methods. On the GPU, the parallel framework achieves an additional fourfold speedup for frame sizes of 32×32 and larger. For 16×16 frames, however, GPU

performance is comparable to the CPU due to the relatively small workload per kernel launch and the associated communication overhead between the CPU and GPU[64,65].

Figure 4b illustrates how the per-iteration time for each method scales with the number of frames analyzed. For this test, we simulated a dataset containing two emitting particles across 1000 frames of size 50×50. Subsets of this dataset were then analyzed, ranging from a single frame to the full 1000-frame stack. Due to its design, the serial framework cannot process datasets with only one frame, rendering it unsuitable for localization tasks. As shown in Fig. 4b and consistent with previous results, the parallel framework achieves approximately a 20-fold speedup on the CPU. However, a computational advantage is only evident for datasets with more than 20 frames using a GPU. For smaller datasets, the communication overhead between CPU and GPU again dominates.

In the following test, we evaluate how the per-iteration time for each method scales with the number of emitting particles. We begin by simulating a dataset of ten 128×128 frames containing a single emitting particle. We then incrementally increase the number of emitting particles up to 200. As shown in Fig. 4c, the per-iteration time of both the serial and the parallel frameworks on the CPU remains relatively stable until the number of emitting particles exceeds 20, beyond which computational cost begins to rise more noticeably. In contrast, the parallel framework running on the GPU remains largely unaffected by the increasing number of emitting particles.

This scaling complexity arises from the use of nonparametric structures, which inherently complicates posterior sampling in physics-inspired frameworks[29,66–68]. In the proposed parallel SPT framework and its serial predecessor, a fixed, user-specified number of candidate particles is assumed, only a subset of which are true emitting particles. The remaining candidates function as "invisible" placeholders. Each candidate is assigned a binary indicator, or load, that determines whether it is active (emitting) or inactive. Since this assignment is updated for all candidates at every iteration, the total number of particles, not just the number of active ones, becomes a key driver of computational cost.

The benefits of combining nonparametric posterior structures with efficient sampling strategies, central to our parallel physics-inspired framework, are further underscored by the results in Fig. 4d, where we fix the number of emitting particles to one while increasing the total number of candidate particles from 1 to 200. As expected, Fig. 4d reveals a more regular scaling behavior consistent with the underlying algorithmic design. Two points are worth noting: first, the original serial framework does not allow the number of candidate particles to be set below five, limiting its flexibility in this regime. Second, with only one emitting particle, the parallel framework exhibits minimal differences in computational efficiency between CPU and GPU implementations. This highlights that GPU parallelism becomes increasingly beneficial only as the number of emitting particles increases.

2.3 Nonparametric cost

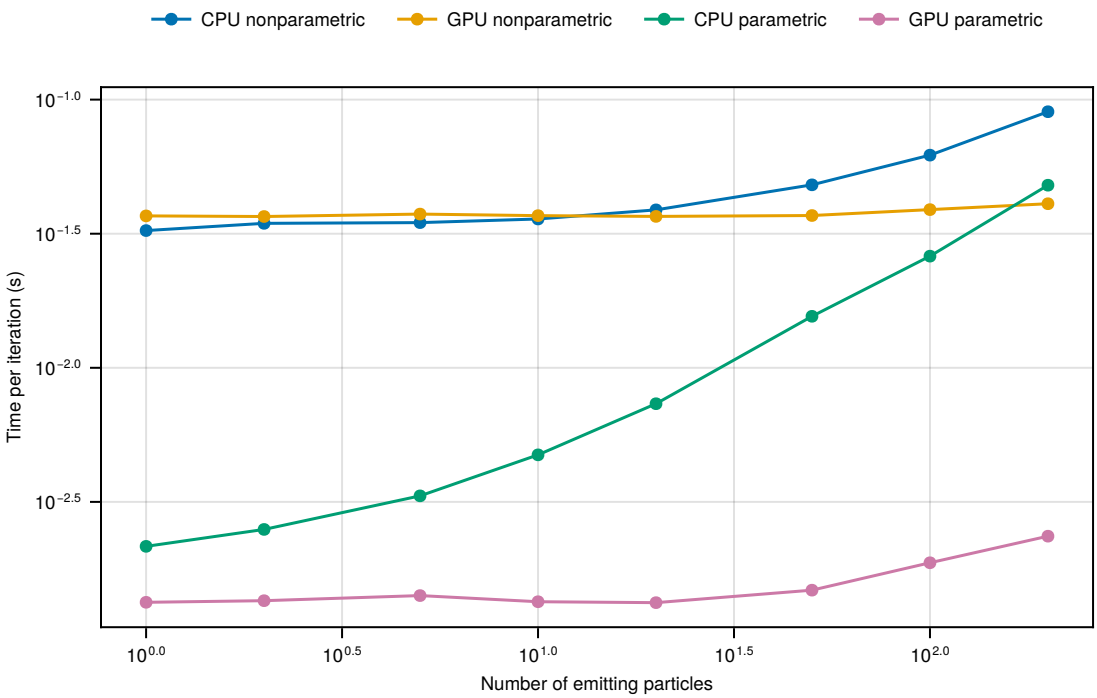

Figure 5: Trade-off between nonparametric flexibility and computational efficiency in our parallel physics-inspired framework for different numbers of emitting particles, illustrated on both CPU and GPU. We compare the performance of the parallel framework in its nonparametric (200 particle candidates, the number of emitting particles ranges from 1 to 200) and parametric (number of candidates equals number of emitting particles) modes, each run on both CPU and GPU. These results highlight a fundamental trade-off: nonparametric formulations offer greater modeling flexibility when the number of particles is unknown or time-varying, but at increased computational cost. The specific needs and constraints of the application should guide the choice between modes. All benchmarks were conducted on the same desktop computer with Intel i7-7700K CPU and Nvidia GTX 1060 6 GB GPU, see Computational efficiency measurement.

Having established the importance of nonparametric structures in physics-inspired frameworks and their role in increasing the complexity of posterior sampling, we now quantify the computational cost associated with this modeling flexibility. To this end, we compare two configurations of the proposed parallel framework: one operating in a nonparametric mode with 200 candidate particles (as in earlier sections), and another in a parametric mode where the number of candidate particles is set equal to the number of true emitters. Both configurations are evaluated on CPU and GPU to assess how hardware affects computational performance under different model assumptions. As shown in Fig. 5, the parametric

mode consistently outperforms the nonparametric mode in computational efficiency on both platforms. Notably, for datasets with 200 emitters, the GPU-accelerated parametric version achieves an additional 20-fold speedup over its nonparametric counterpart.

This result highlights an important trade-off: while nonparametric models offer greater flexibility and robustness, especially in settings where the number of particles is unknown, they also come with increased computational cost. The choice between parametric and nonparametric formulations should thus be guided by the specific demands of the application, balancing the need for modeling generality against available computational resources.

3 Discussion

Integrating realistic physics into data analysis through complex likelihood functions introduces a dramatic increase in computational cost. While embarrassingly parallel methods have been developed to distribute these computations across large CPU clusters[18–20,22,46–50], such approaches are most effective when the underlying inference task requires little to no communication between processes. However, this assumption breaks down when modeling systems governed by tightly coupled physical constraints—such as time-correlated motion in particle tracking—where frequent communication between frames or model components is essential. In these scenarios, accelerating the performance of a single inference chain remains a critical bottleneck.

To address this challenge in the context of SPT, we introduce BNP-Track 2.0, a parallel, physics-inspired framework that takes a fundamentally different approach toward computational efficiency. Instead of distributing independent MCMC chains or partitioning datasets across nodes, this parallel framework exploits the fine-grained parallelism embedded within the likelihood structure to speed up a single inference chain. By parallelizing both the likelihood evaluation and associated sampling updates, the framework leverages modern desktop-class hardware with a single GPU to achieve substantial speedups, eliminating the need for large-scale computing infrastructure while retaining the full statistical rigor of Bayesian inference. Importantly, our parallel physics-inspired framework preserves its serial counterpart's high accuracy and data efficiency, while achieving a nearly 50-fold improvement in computational speed. This substantial performance gain transforms the framework from a powerful but time-intensive tool into one capable of supporting high-throughput analyses. This parallel framework extends the practical reach of likelihood-based inference in analyzing complex spatiotemporal data by bridging the gap between physical rigor and computational efficiency.

As our proposed framework does not rely on embarrassingly parallel strategies, it remains fully compatible with them. Such independence means the framework can be further optimized by incorporating embarrassingly parallel techniques, such as running multiple inference chains or distributing data across nodes, potentially unlocking even greater computational efficiency without compromising its core design. As a result, the framework is well-positioned to scale beyond desktop environments and tackle large-scale data analysis tasks, such as processing the massive volumetric datasets produced by lattice light-sheet

microscopy[69].

Beyond hardware improvements, such as utilizing TPUs for enhanced performance, several emerging computational techniques could further optimize our parallel framework. We highlight three promising directions: (1) Mixed precision computing[70–72], which allows our framework to treat different variables with different precision levels. For instance, object positions can be stored at lower precision, while probability density estimates of track configurations can be maintained at higher precision. This strategy reduces memory usage and accelerates computation without sacrificing accuracy. (2) Randomized numerical linear algebra[73], which can speed up matrix operations by leveraging stochastic approximations instead of exact computations, further improving the scalability of our parallel framework in high-dimensional tracking problems. (3) Deep learning-based acceleration[25,74,75], which can be employed to approximate computational bottlenecks in physics-inspired inference. For instance, deep architectures such as normalizing flows[76] or diffusion models[77,78] can be trained to approximate complex likelihoods or to propose efficient samples from challenging posterior distributions[74].

By offering concrete strategies for principled, non-naive parallelization that preserves mathematical rigor and statistical accuracy, we hope to encourage the adoption of physics-inspired likelihood-based data analysis methods equipped with the parallelism provided by modern computing architectures. Through using techniques such as vectorization, multi-threading, and GPU acceleration, we are no longer limited to either small and simple systems or systems where likelihood can be embarrassingly parallelized[18,19,22,46–50]. Moreover, thanks to the development of modern programming languages and libraries[79–83], scientists can now implement these acceleration techniques with much lower entry barriers.

We envision a growing role for scalable, interpretable, and parallel likelihood-based frameworks that match the speed of black-box or approximated models and retain transparency and physical grounding. As data grows in volume and complexity, bridging the gap between statistical rigor and computational scalability will be crucial. BNP-Track 2.0 represents one step toward this goal, offering a viable solution for tracking fast, noisy, and crowded biological systems without compromising accuracy or data efficiency.

Acknowledgments

S.P. acknowledges support from the NIH (R35GM148237), ARO (W911NF-23-1-0304), and NSF (Grant No. 2310610).

Data availability

All data supporting the findings of this study are available as benchmark datasets included with the accompanying code repository.

Code availability

Our code is publicly available at `https://github.com/LabPresse/SP2T.jl`.

Competing interests

The authors declare no competing interests.

4 Methods

This section introduces the core parallelization and optimization strategies employed in the proposed parallel framework. For clarity, we define the following notation: let N denote the number of frames; I and J represent the number of pixels in the x- and y-directions per frame, respectively; and w_n^{ij} denote the binary measurement collected at pixel (i, j) in frame n. Let M be the number of emitting particles, with $\mathbf{x}_n^m$ representing the position of particle m at frame n. The complete dataset can then be written as $w_{1:N}^{1:I,1:J} = \{w_n^{ij} | n = 1, \ldots, N; , i = 1, \ldots, I; , j = 1, \ldots, J\}$, and the full set of particle tracks as $\mathbf{x}_{1:N}^{1:M}$.

As discussed in the Introduction, the goal of tracking is to establish a full probability distribution over M (the number of particles) and $\mathbf{x}_{1:N}^{1:M}$ (all elements of all tracks), given $w_{1:N}^{1:I,1:J}$ (the recorded frames). In addition, we will also need to estimate the mean squared displacement (MSD) of the particles, as it is typically unknown *a priori*. Using the notations introduced so far, we can express this target probability distribution as $\mathbb{P}\left(M, \mathbf{x}_{1:N}^{1:M}, \mathrm{MSD} \middle| w_{1:N}^{1:I,1:J}\right)$.

In the Bayesian paradigm, this target probability distribution is termed the posterior distribution, representing the probability distribution of all variables of interest given the collected data. To construct this posterior, we apply Bayes' theorem and get

$$\mathbb{P}\left(M, \mathbf{x}_{1:N}^{1:M}, \mathrm{MSD} \middle| w_{1:N}^{1:I,1:J}\right) = \frac{\mathbb{P}\left(w_{1:N}^{1:I,1:J} \middle| M, \mathbf{x}_{1:N}^{1:M}, \mathrm{MSD}\right) \mathbb{P}\left(M, \mathbf{x}_{1:N}^{1:M}, \mathrm{MSD}\right)}{\mathbb{P}\left(w_{1:N}^{1:I,1:J}\right)}. \tag{1}$$

On the right hand side, the first term in the numerator, $\mathbb{P}\left(w_{1:N}^{1:I,1:J} \middle| M, \mathbf{x}_{1:N}^{1:M}, \mathrm{MSD}\right)$, represents the likelihood, which is the probability density of the data given the variables of interest. This term usually describes the system's physics (the model for photon detection in this context) and realistic complications such as hot pixels. The second term, $\mathbb{P}\left(M, \mathbf{x}_{1:N}^{1:M}, \mathrm{MSD}\right)$, known as the prior, imposes additional constraints on the variables of interest, such as a particle's motion model and ensuring the MSD remains non-negative. The denominator, $\mathbb{P}\left(w_{1:N}^{1:I,1:J}\right)$, is referred to as the evidence and is generally treated as a normalization constant, as it does not depend on any variable of interest.

Our parallel framework employs Gibbs sampling[29,84] to infer the variables of interest. This iterative procedure updates each variable by sampling from its conditional posterior

distribution, holding the others fixed. Specifically, the particle trajectories $\mathbf{x}_{1:N}^{1:M}$ are first updated from the conditional posterior $\mathbb{P}\left(\mathbf{x}_{1:N}^{1:M}\middle| M, \mathrm{MSD}, w_{1:N}^{1:I,1:J}\right)$, followed by updating the number of emitting particles M via $\mathbb{P}\left(M\middle|\mathbf{x}_{1:N}^{1:M}, \mathrm{MSD}, w_{1:N}^{1:I,1:J}\right)$, and finally, the mean squared displacement MSD is updated using $\mathbb{P}\left(\mathrm{MSD}\middle|\mathbf{x}_{1:N}^{1:M}, M, w_{1:N}^{1:I,1:J}\right)$.

4.1 Likelihood parallelization

As introduced above, the likelihood $\mathbb{P}\left(w_{1:N}^{1:I,1:J}\middle| M, \mathbf{x}_{1:N}^{1:M}, \mathrm{MSD}\right)$ represents the probability distribution of the data given the number of particles, the particle tracks, and the MSD. Since it describes the system's physics, it is usually the most computationally expensive term in the posterior distribution. Therefore, we first focus on parallelizing this term to improve the overall computation time of the framework.

To express this likelihood mathematically, we first observe that once all particle tracks are known, neither the number of particles nor the MSD affects photon detection. This observation allows us to simplify $\mathbb{P}\left(w_{1:N}^{1:I,1:J}\middle| M, \mathbf{x}_{1:N}^{1:M}, \mathrm{MSD}\right)$ to $\mathbb{P}\left(w_{1:N}^{1:I,1:J}\middle|\mathbf{x}_{1:N}^{1:M}\right)$.

Furthermore, considering that we are dealing with fluorescently emitting particles whose lifetimes (on the order of nanoseconds[85]) are significantly shorter than the frame period (microseconds and above), the probability that an emitting particle excited during the exposure period of one frame will result in a photon detection in the next frame is negligible. Therefore, it is reasonable to assume that particle positions at frame n do not influence other frames. This assumption allows us to further simplify the likelihood as a product:

$$\mathbb{P}\left(w_{1:N}^{1:I,1:J}\middle|\mathbf{x}_{1:N}^{1:M}\right) = \prod_{n=1}^{N} \mathbb{P}\left(w_n^{1:I,1:J}\middle|\mathbf{x}_n^{1:M}\right). \tag{2}$$

Similarly, since the pixels of a detector usually operate independently, we can express the per-frame likelihood $\mathbb{P}\left(w_n^{1:I,1:J}\middle|\mathbf{x}_n^{1:M}\right)$ as a product over the individual pixel likelihoods:

$$\mathbb{P}\left(w_n^{1:I,1:J}\middle|\mathbf{x}_n^{1:M}\right) = \prod_{p=1}^{P} \mathbb{P}\left(w_n^{ij}\middle|\mathbf{x}_n^{1:M}\right). \tag{3}$$

The per-pixel likelihood $\mathbb{P}\left(w_n^{ij}\middle|\mathbf{x}_n^{1:M}\right)$ is influenced by four key factors: the point spread function (PSF) of the optical system, the pixel area A_{ij}, the brightness of the particles h, and the number of emitting particles M. The PSF characterizes how light from a point source is distributed across the detector[35]. At the same time, brightness represents the expected number of photons arriving at a pixel perfectly aligned with the PSF of a particle located at the focal plane. The total number of photons received by pixel p from all emitting particles, denoted as u_n^{ij}, can be expressed as:

$$u_n^{ij} = h \sum_{m=1}^{M} \iint_{A_{ij}} \mathrm{PSF}\left(q_x, q_y; \mathbf{x}_n^m\right) \mathrm{d}q_x \, \mathrm{d}q_y. \tag{4}$$

Then $\mathbb{P}\left(w_n^{ij}\middle|\mathbf{x}_n^{1:M}\right)$ can be constructed using equation (4) and $\mathbb{P}\left(w_n^{ij}|u_n^{ij}\right)$, which is specific to the detector and the experimental conditions. For example, if the detector is a single-photon avalanche diode (SPAD) array, $\mathbb{P}\left(w_n^{ij}|u_n^{ij}\right)$ can be modeled as a Bernoulli distribution with a probability of $1-\exp\left(-u_n^{ij}\right)$. In contrast, if the detector is a charge-coupled device (CCD), $\mathbb{P}\left(w_n^{ij}|u_n^{ij}\right)$ can be modeled as a Gaussian distribution[35].

In equation (4), the form of the PSF is often approximated as a Gaussian function of[35]:

$$\mathrm{PSF}\left(q_x, q_y; \mathbf{x}_n^m\right) \approx C_n^m \exp\left(-\frac{\left(q_x - x_n^m\right)^2 + \left(q_y - y_n^m\right)^2}{2v_n^m}\right), \tag{5}$$

where C_n^m is some normalization constant and v_n^m is the variance of the Gaussian function in the lateral direction, which is dependent on the emission wavelength, numerical aperture, and a particle's z-location. With this approximation, equation (4) can be written as a matrix multiplication:

$$u_n^{ij} \approx h \sum_{m=1}^{M} u_{n,x}^{i,m} u_{n,y}^{j,m}. \tag{6}$$

Here, $u_{n,x}^{p,m} = \int \sqrt{C_n^m} \exp\left(-\frac{(q_x - x_n^m)^2}{2v_n^m}\right) \mathrm{d}q_x$ which can be calculated using the error function. It follows that the expected photon count at all pixels on frame n can be expressed as a matrix multiplication:

$$u_n^{1:I,1:J} \approx h U_{n,x} U_{n,y}^T, \tag{7}$$

where $U_{n,x}$ is an $I \times M$ matrix with $u_{n,x}^{p,m}$ being its elements. To calculate $u_{1:N}^{1:I,1:J}$, we can perform matrix multiplications for all N frames in parallel. This operation is known as batched matrix multiplication. It is implemented in many numerical and deep-learning libraries[81–83,86–89]. Similarly, as all elements in $u_{1:N}^{1:I,1:J}$ are calculated, each $\mathbb{P}\left(w_n^{ij}|u_n^{ij}\right)$ can be calculated in parallel as well using techniques such as broadcasting[90–92].

4.2 Two-step track update

As described above, particle trajectories are updated by sampling from the conditional posterior distribution $\mathbb{P}\left(\mathbf{x}_{1:N}^{1:M}\middle|M, \mathrm{MSD}, w_{1:N}^{1:I,1:J}\right)$ within the Bayesian framework. Based on the concept of Markov boundary[29,93], the minimal set of random variables containing all useful information to infer another random variable, our parallel framework achieves this through a two-step update scheme designed to maximize parallel efficiency. First, we compute the conditional distribution $\mathbb{P}\left(\mathbf{x}_{1:2:N}^{1:M}\middle|\mathbf{x}_{2:2:N}^{1:M}, M, \mathrm{MSD}, w_{1:N}^{1:I,1:J}\right)$, where 1:2:$N$ and 2:2:N denote all odd and even frame indices, respectively. By holding the particle positions at all even frames fixed, the positions at odd frames become conditionally independent and can thus be updated in parallel. In the second step, we compute the conditional distribution for the even indices, $\mathbb{P}\left(\mathbf{x}_{2:2:N}^{1:M}\middle|\mathbf{x}_{1:2:N}^{1:M}, M, \mathrm{MSD}, w_{1:N}^{1:I,1:J}\right)$, and update all even frame positions in parallel while holding the odd ones fixed. This alternating blockwise update strategy allows all emitting particle tracks to be refreshed in just two passes, each enabling full parallelization across frames.

4.3 Memory management

In addition to leveraging parallelism, our framework enhances computational efficiency through careful memory management. A key optimization involves ensuring contiguous memory allocation for the most computationally intensive components of the algorithm[65,94,95]. In particular, the likelihood term, $\mathbb{P}\left(w_{1:N}^{1:I,1:J} \middle| \mathbf{x}_{1:N}^{1:M}\right)$, is computed using only the M emitting particles. Our parallel framework dynamically reorganizes particle tracks to optimize this step so that the data associated with emitting particles is stored in a contiguous memory block.

Another key memory management strategy employed in our framework is minimizing data copying and avoiding the overhead associated with garbage collection—reclaiming memory no longer in use[94,96]. To achieve this, our framework preallocates memory for intermediate variables—such as $u_{1:N}^{1:I,1:J}$—at the start of execution and reuses this memory throughout the computation.

Similarly, to minimize communication overhead between the CPU and GPU, our framework is designed so that large data transfers occur only once at the beginning of execution. Only lightweight scalar values are exchanged between the CPU and GPU during runtime, significantly reducing latency and maximizing computational throughput.

4.4 Computational efficiency measurement

All the tests presented in this study were performed on a desktop computer equipped with an Intel i7-7700K CPU (four cores, eight threads, up to 4.5 GHz), an Nvidia GTX 1060 6 GB GPU, 64 GB of RAM, and Ubuntu 24.04.2 as the operating system. To evaluate the per-iteration runtime of BNP-Track[30] and the proposed parallel framework, each method was executed through an initial test run to account for and complete any initialization overhead. Subsequently, the runtime for 1000 iterations was measured, and the reported per-iteration time was computed by dividing the total runtime by 1000.

4.5 Software

Our framework is implemented in Julia[79], a high-level, high-performance dynamic programming language designed for technical computing. The framework leverages several core Julia packages: CUDA.jl[80] for GPU acceleration on Nvidia hardware, Makie.jl[97] for high-quality data visualization, Distributions.jl[98] for probabilistic modeling and working with statistical distributions, and Flux.jl[83,86] for efficiently implementing batched matrix operations and numerical routines.

References

1. Jazani, S., Sgouralis, I. & Pressé, S. A method for single molecule tracking using a conventional single-focus confocal setup. *J. Chem. Phys.* **150.** ISSN: 1089-7690 (2019).

2. Jazani, S. *et al.* An alternative framework for fluorescence correlation spectroscopy. *Nat. Commun.* **10,** 3662. ISSN: 2041-1723 (2019).

3. Tavakoli, M. *et al.* Pitching Single-Focus Confocal Data Analysis One Photon at a Time with Bayesian Nonparametrics. *Phys. Rev. X* **10,** 011021. ISSN: 2160-3308 (2020).

4. Karslake, J. D. *et al.* SMAUG: Analyzing single-molecule tracks with nonparametric Bayesian statistics. *Methods* **193,** 16. ISSN: 1046-2023 (2021).

5. Van de Schoot, R. *et al.* Bayesian statistics and modelling. *Nat. Rev. Methods Primers* **1.** ISSN: 2662-8449 (2021).

6. Kilic, Z. *et al.* Extraction of rapid kinetics from smFRET measurements using integrative detectors. *Cell Rep. Phys. Sci.* **2,** 100409. ISSN: 2666-3864 (2021).

7. Kilic, Z., Sgouralis, I. & Pressé, S. Generalizing HMMs to Continuous Time for Fast Kinetics: Hidden Markov Jump Processes. *Biophys. J.* **120,** 409. ISSN: 0006-3495 (2021).

8. Vroylandt, H., Goudenège, L., Monmarché, P., Pietrucci, F. & Rotenberg, B. Likelihood-based non-Markovian models from molecular dynamics. *Proc. Natl. Acad. Sci. U.S.A.* **119.** ISSN: 1091-6490 (2022).

9. Wu, Y.-L. *et al.* Maximum-likelihood model fitting for quantitative analysis of SMLM data. *Nat. Methods* **20,** 139. ISSN: 1548-7105 (2022).

10. Linka, K. *et al.* Bayesian Physics Informed Neural Networks for real-world nonlinear dynamical systems. *Comput. Methods Appl. Mech. Engrg.* **402,** 115346. ISSN: 0045-7825 (2022).

11. Saurabh, A., Fazel, M., Safar, M., Sgouralis, I. & Pressé, S. Single-photon smFRET. I: Theory and conceptual basis. en. *Biophys. Rep.* **3,** 100089. ISSN: 2667-0747 (2023).

12. Saurabh, A., Safar, M., Fazel, M., Sgouralis, I. & Pressé, S. Single-photon smFRET: II. Application to continuous illumination. en. *Biophys. Rep.* **3,** 100087. ISSN: 2667-0747 (2023).

13. Safar, M. *et al.* Single-photon smFRET. III. Application to pulsed illumination. en. *Biophys. Rep.* **2,** 100088. ISSN: 2667-0747 (2022).

14. Gopich, I. V., Louis, J. M. & Chung, H. S. Maximum Likelihood Analysis of Diffusing Molecules with Conformational Dynamics in Single-Molecule FRET. *J. Phys. Chem. B* **129,** 2187. ISSN: 1520-5207 (2025).

15. Jurich, C., Shao, Q., Ran, X. & Yang, Z. J. Physics-based modeling in the new era of enzyme engineering. *Nat. Comput. Sci.* **5,** 279. ISSN: 2662-8457 (2025).

16. Fan, C. *et al.* A likelihood-based framework for demographic inference from genealogical trees. *Nat. Genet.* **57,** 865. ISSN: 1546-1718 (2025).

17. Xu, L. W., Jazani, S., Kilic, Z. & Pressé, S. Single-molecule reaction-diffusion. *Biophys. J.* ISSN: 0006-3495 (2025).

18. Wang, X. & Dunson, D. B. *Parallelizing MCMC via Weierstrass Sampler* 2014. arXiv: `1312.4605 [stat.CO]`.

19. Wang, X., Guo, F., Heller, K. A. & Dunson, D. B. *Parallelizing MCMC with Random Partition Trees* in *Advances in Neural Information Processing Systems* (eds Cortes, C., Lawrence, N., Lee, D., Sugiyama, M. & Garnett, R.) **28** (Curran Associates, Inc., 2015). `https://proceedings.neurips.cc/paper_files/paper/2015/file/40008b9a5380fcacce3976bf7c08af5b-Paper.pdf`.

20. Scott, S. L. *et al.* Bayes and big data: the consensus Monte Carlo algorithm. *Int. J. Manage. Sci. Eng. Manage.* **11,** 78. ISSN: 1750-9661 (2016).

21. Nemeth, C. & Sherlock, C. Merging MCMC Subposteriors through Gaussian-Process Approximations. *Bayesian Anal.* **13.** ISSN: 1936-0975 (2018).

22. Srivastava, S., Li, C. & Dunson, D. B. Scalable Bayes via Barycenter in Wasserstein Space. *J. Mach. Learn. Res.* **19,** 1. `http://jmlr.org/papers/v19/17-084.html` (2018).

23. Mesquita, D., Blomstedt, P. & Kaski, S. *Embarrassingly Parallel MCMC using Deep Invertible Transformations* in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference* (eds Adams, R. P. & Gogate, V.) **115** (PMLR, 2020), 1244. `https://proceedings.mlr.press/v115/mesquita20a.html`.

24. Chen, X., Cheng, J. Q. & Xie, M.-g. *Divide-and-conquer methods for big data analysis* 2021.

25. Winter, S., Campbell, T., Lin, L., Srivastava, S. & Dunson, D. B. Emerging Directions in Bayesian Computation. *Stat. Sci.* **39,** 62. ISSN: 0883-4237 (2024).

26. Adrian, R. J. & Westerweel, J. *Particle image velocimetry Cambridge aerospace series* **30.** Literaturverz. S. 529 - 546. 558 pp. ISBN: 9780521440080 (Cambridge Univ. Press, Cambridge [u.a.], 2011).

27. Wiltschko, A. B. *et al.* Mapping Sub-Second Structure in Mouse Behavior. *Neuron* **88,** 1121. ISSN: 0896-6273 (2015).

28. Li, Y. *Assessing Speech, Action, and Temporal Synchrony in Parent-Child Play With Familiar and Unfamiliar Objects* English. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2025-05-12. PhD thesis (Arizona State University, 2025), 159.

29. Pressé, S. & Sgouralis, I. *Data Modeling for the Sciences: Applications, Basics, Computations* ISBN: 9781009098502 (Cambridge University Press, 2023).

30. Sgouralis, I. *et al.* BNP-Track: a framework for superresolved tracking. *Nat. Methods* **21,** 1716. ISSN: 1548-7105 (2024).

31. Tinevez, J.-Y. *et al.* TrackMate: An open and extensible platform for single-particle tracking. *Methods* **115,** 80. ISSN: 1046-2023 (2017).

32. Chenouard, N. *et al.* Objective comparison of particle tracking methods. *Nat. Methods* **11,** 281. ISSN: 1548-7105 (2014).

33. Jaqaman, K. *et al.* Robust single-particle tracking in live-cell time-lapse sequences. *Nat. Methods* **5,** 695. ISSN: 1548-7105 (2008).

34. Lelek, M. *et al.* Single-molecule localization microscopy. *Nat. Rev. Methods Primers* **1,** 39. ISSN: 2662-8449 (2021).

35. Fazel, M. *et al.* Fluorescence microscopy: A statistics-optics perspective. *Rev. Mod. Phys.* **96,** 025003. ISSN: 1539-0756 (2 2024).

36. Balasubramanian, H., Hobson, C. M., Chew, T.-L. & Aaron, J. S. Imagining the future of optical microscopy: everything, everywhere, all at once. *Commun. Biol.* **6,** 1096. ISSN: 2399-3642 (2023).

37. Qiao, C. *et al.* Evaluation and development of deep neural networks for image super-resolution in optical microscopy. *Nat. Methods* **18,** 194. ISSN: 1548-7105 (2021).

38. Song, D. *et al.* Deep Learning-Assisted Automated Multidimensional Single Particle Tracking in Living Cells. *Nano Lett.* **24,** 3082. ISSN: 1530-6992 (2024).

39. Qiao, C. *et al.* A neural network for long-term super-resolution imaging of live cells with reliable confidence quantification. *Nat. Biotechnol.* ISSN: 1546-1696 (2025).

40. Qiao, C. *et al.* Fast-adaptive super-resolution lattice light-sheet microscopy for rapid, long-term, near-isotropic subcellular imaging. *Nat. Methods* **22,** 1059. ISSN: 1548-7105 (2025).

41. Zhong, Y., Li, C., Zhou, H. & Wang, G. Developing Noise-Resistant Three-Dimensional Single Particle Tracking Using Deep Neural Networks. *Anal. Chem.* **90,** 10748. ISSN: 1520-6882 (2018).

42. Spilger, R. *et al.* A Recurrent Neural Network for Particle Tracking in Microscopy Images Using Future Information, Track Hypotheses, and Multiple Detections. *IEEE Trans. Image Process.* **29,** 3681. ISSN: 1941-0042 (2020).

43. Yao, Y., Smal, I., Grigoriev, I., Akhmanova, A. & Meijering, E. Deep-learning method for data association in particle tracking. *Bioinformatics* **36** (ed Wren, J.) 4935. ISSN: 1367-4811 (2020).

44. Spilger, R. *et al.* Deep probabilistic tracking of particles in fluorescence microscopy images. en. *Med. Image Anal.* **72,** 102128 (2021).

45. Cheng, H.-J., Hsu, C.-H., Hung, C.-L. & Lin, C.-Y. A review for cell and particle tracking on microscopy images using algorithms and deep learning technologies. *Biomed. J.* **45,** 465. ISSN: 2319-4170 (2022).

46. Swendsen, R. H. & Wang, J.-S. Replica Monte Carlo Simulation of Spin-Glasses. *Phys. Rev. Lett.* **57,** 2607. ISSN: 0031-9007 (21 1986).

47. Earl, D. J. & Deem, M. W. Parallel tempering: Theory, applications, and new perspectives. *Phys. Chem. Chem. Phys.* **7,** 3910. ISSN: 1463-9084 (23 2005).

48. Goodman, J. & Weare, J. Ensemble samplers with affine invariance. *Commun. Appl. Math. Comput. Sci.* **5,** 65. ISSN: 1559-3940 (2010).

49. Jacob, P. E., O' Leary, J. & Atchadé, Y. F. Unbiased Markov Chain Monte Carlo Methods with Couplings. *J. R. Stat. Soc. Ser. B Stat. Method.* **82,** 543. ISSN: 1467-9868. eprint: `https://academic.oup.com/jrsssb/article-pdf/82/3/543/49324120/jrsssb_82_3_543.pdf` (2020).

50. Glatt-Holtz, N. E., Holbrook, A. J., Krometis, J. A. & Mondaini, C. F. Parallel MCMC algorithms: theoretical foundations, algorithm design, case studies. *Trans. Math. Appl.* **8,** tnae004. ISSN: 2398-4945. eprint: `https://academic.oup.com/imatrm/article-pdf/8/2/tnae004/60767276/tnae004.pdf` (2024).

51. Vaughan, J. C., Jia, S. & Zhuang, X. Ultrabright photoactivatable fluorophores created by reductive caging. *Nat. Methods* **9,** 1181. ISSN: 1548-7105 (2012).

52. Jonkman, J., Brown, C. M., Wright, G. D., Anderson, K. I. & North, A. J. Tutorial: guidance for quantitative confocal microscopy. *Nat. Protoc.* **15,** 1585. ISSN: 1750-2799 (2020).

53. Galvanetto, N. *et al.* Extreme dynamics in a biomolecular condensate. *Nature* **619,** 876. ISSN: 1476-4687 (2023).

54. Bi, G. *et al.* The ZAR1 resistosome is a calcium-permeable channel triggering plant immune signaling. *Cell* **184,** 3528. ISSN: 0092-8674 (2021).

55. Ahn, J. H. *et al.* Phase separation drives aberrant chromatin looping and cancer development. *Nature* **595,** 591. ISSN: 1476-4687 (2021).

56. Folkmann, A. W., Putnam, A., Lee, C. F. & Seydoux, G. Regulation of biomolecular condensates by interfacial protein clusters. *Science* **373,** 1218. ISSN: 1095-9203 (2021).

57. Victoria, A. H. & Maragatham, G. Automatic tuning of hyperparameters using Bayesian optimization. *Evol. Syst.* **12,** 217. ISSN: 1868-6486 (2020).

58. Zhao, S., Li, K., Ahn, C. K., Huang, B. & Liu, F. Tuning-Free Bayesian Estimation Algorithms for Faulty Sensor Signals in State-Space. *IEEE Trans. Ind. Electron.* **70,** 921. ISSN: 1557-9948 (2023).

59. Smith, C. S. *et al.* An automated Bayesian pipeline for rapid analysis of single-molecule binding data. *Nat. Commun.* **10.** ISSN: 2041-1723 (2019).

60. Bryan IV, J. S., Sgouralis, I. & Pressé, S. Diffraction-limited molecular cluster quantification with Bayesian nonparametrics. *Nat. Comput. Sci.* **2,** 102. ISSN: 2662-8457 (2022).

61. Kilic, Z., Schweiger, M., Moyer, C., Shepherd, D. & Pressé, S. Gene expression model inference from snapshot RNA data using Bayesian non-parametrics. en. *Nat. Comput. Sci.* **3,** 174. ISSN: 2662-8457 (2023).

62. Chen, Z., Biggie, H., Ahmed, N., Julier, S. & Heckman, C. Kalman Filter Auto-Tuning With Consistent and Robust Bayesian Optimization. *IEEE Trans. Aerosp. Electron. Syst.* **60,** 2236. ISSN: 2371-9877 (2024).

63. Saurabh, A. *et al.* Approaching maximum resolution in structured illumination microscopy via accurate noise modeling. *npj Imaging* **3.** ISSN: 2948-197X (2025).

64. Gregg, C. & Hazelwood, K. *Where is the data? Why you cannot debate CPU vs. GPU performance without the answer* in *(IEEE ISPASS) IEEE INTERNATIONAL SYMPOSIUM ON PERFORMANCE ANALYSIS OF SYSTEMS AND SOFTWARE* (IEEE, 2011), 134.

65. Hennessy, J. L. & Patterson, D. A. *Computer architecture. A quantitative approach* Sixth edition (eds Patterson, D. A. & Asanović, K.) Literaturverzeichnis: Seite R-1-R-36, 936. ISBN: 9780128119068 (Elsevier Science & Technology Books, Cambridge, Mass., 2017).

66. Beal, M., Ghahramani, Z. & Rasmussen, C. *The Infinite Hidden Markov Model* in *Advances in Neural Information Processing Systems* (eds Dietterich, T., Becker, S. & Ghahramani, Z.) **14** (MIT Press, 2001). `https://proceedings.neurips.cc/paper_files/paper/2001/file/e3408432c1a48a52fb6c74d926b38886-Paper.pdf`.

67. Teh, Y. W., Jordan, M. I., Beal, M. J. & Blei, D. M. Hierarchical Dirichlet Processes. *J. Amer. Statistical Assoc.* **101,** 1566. ISSN: 1537-274X (2006).

68. Müller, P., Quintana, F. A., Jara, A. & Hanson, T. *Bayesian Nonparametric Data Analysis* ISBN: 9783319368429 (Springer International Publishing AG, 2016).

69. Ruan, X. *et al.* Image processing tools for petabyte-scale light sheet microscopy data. *Nat. Methods* **21,** 2342. ISSN: 1548-7105 (2024).

70. Higham, N. J. & Mary, T. Mixed precision algorithms in numerical linear algebra. *Acta Numer.* **31,** 347. ISSN: 1474-0508 (2022).

71. Micikevicius, P. *et al. Mixed Precision Training* 2018. arXiv: `1710.03740 [cs.AI]`.

72. Wang, L. *et al. Ladder: Enabling Efficient Low-Precision Deep Learning Computing through Hardware-aware Tensor Transformation* in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)* (USENIX Association, Santa Clara, CA, 2024), 307. ISBN: 978-1-939133-40-3. `https://www.usenix.org/conference/osdi24/presentation/wang-lei`.

73. Murray, R. *et al. Randomized Numerical Linear Algebra : A Perspective on the Field With an Eye to Software* 2023. arXiv: `2302.11474 [math.NA]`.

74. Gabrié, M., Rotskoff, G. M. & Vanden-Eijnden, E. Adaptive Monte Carlo augmented with normalizing flows. *Proc. Natl. Acad. Sci. U.S.A.* **119,** e2109420119. ISSN: 1091-6490 (2022).

75. Asghar, S., Pei, Q.-X., Volpe, G. & Ni, R. Efficient rare event sampling with unsupervised normalizing flows. *Nat. Mach. Intell.* **6,** 1370. ISSN: 2522-5839 (2024).

76. Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S. & Lakshminarayanan, B. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research* **22,** 1. `http://jmlr.org/papers/v22/19-1028.html` (2021).

77. Croitoru, F.-A., Hondru, V., Ionescu, R. T. & Shah, M. Diffusion Models in Vision: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **45,** 10850. ISSN: 1939-3539 (2023).

78. Yang, L. *et al.* Diffusion Models: A Comprehensive Survey of Methods and Applications. *ACM Comput. Surv.* **56,** 1. ISSN: 1557-7341 (2023).

79. Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. B. Julia: A Fresh Approach to Numerical Computing. *SIAM Rev.* **59,** 65. ISSN: 1095-7200 (2017).

80. Besard, T., Foket, C. & De Sutter, B. Effective Extensible Programming: Unleashing Julia on GPUs. *IEEE Trans. Parallel Distrib. Syst.* **30,** 827. ISSN: 2161-9883 (2019).

81. Ansel, J. *et al. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation* in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (ACM, 2024), 929.

82. Developers, T. *TensorFlow* 2025.

83. Innes, M. *et al. Fashionable Modelling with Flux* 2018.

84. Geman, S. & Geman, D. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6,** 721. ISSN: 0162-8828 (1984).

85. *Fluorescence Quantum Yields and Lifetimes for Alexa Fluor Dyes—Table* `https://www.thermofisher.com/us/en/home/references/molecular-probes-the-handbook/tables/fluorescence-quantum-yields-and-lifetimes-for-alexa-fluor-dyes.html`.

86. Innes, M. Flux: Elegant machine learning with Julia. *J. Open Source Softw.* **3,** 602. ISSN: 2475-9066 (2018).

87. Dongarra, J., Hammarling, S., Higham, N. J., Relton, S. D. & Zounon, M. in *Euro-Par 2017: Parallel Processing* 511 (Springer International Publishing, 2017). ISBN: 9783319642031.

88. Dongarra, J. *et al.* The Design and Performance of Batched BLAS on Modern High-Performance Computing Systems. *Procedia Comput. Sci.* **108,** 495. ISSN: 1877-0509 (2017).

89. Abdelfattah, A. *et al.* A Set of Batched Basic Linear Algebra Subprograms and LAPACK Routines. *ACM Trans. Math. Softw.* **47,** 1. ISSN: 1557-7295 (2021).

90. Quinn, M. J. *Parallel computing. Theory and practice* 2., [rev.] ed., internat. ed., [Nachdr.] Literaturverz. S. 391 - 433. 446 pp. ISBN: 0070512949 (McGraw-Hill, New York [u.a.], 1995).

91. Valiant, L. G. A bridging model for parallel computation. *Commun. ACM* **33,** 103. ISSN: 1557-7317 (1990).

92. *Introduction to parallel computing* 2. ed., [reprint.] (ed Grama, A.) Includes bibliographical references and index. 636 pp. ISBN: 9780201648652 (Pearson, Harlow [u.a.], 2011).

93. Pearl, J. *Probabilistic Reasoning in Intelligent Systems. Networks of Plausible Inference* Rev. 2. print. 584 pp. ISBN: 1558604790 (Elsevier Science, 2014).

94. Drepper, U. *What every programmer should know about memory* `https://people.freebsd.org/~lstewart/articles/cpumemory.pdf` (2025).

95. Goodrich, M. T., Tamassia, R. & Mount, D. M. *Data structures and algorithms in C++* 2. ed. Literaturverz. S. [697] - 701. 714 pp. ISBN: 9780470383278 (Wiley, Hoboken, NJ, 2011).

96. Jones, R., Hosking, A. & Moss, E. *The Garbage Collection Handbook: The Art of Automatic Memory Management* ISBN: 9781003276142 (Chapman and Hall/CRC, 2023).

97. Danisch, S. & Krumbiegel, J. Makie.jl: Flexible high-performance data visualization for Julia. *J. Open Source Softw.* **6,** 3349. ISSN: 2475-9066 (2021).

98. Besançon, M. *et al.* Distributions.jl: Definition and Modeling of Probability Distributions in the JuliaStats Ecosystem. *J. Stat. Softw.* **98.** ISSN: 1548-7660 (2021).