An Accessible Python Framework for Real-Time Magnetic Tweezers Microscope Control and Image Processing

James A. London[1,2,*], Abhishek K. Singh[1,*], Teague C. Svendsen[1], Naciye Esma Tirtom[1], Zachary A. Root[1], and Richard Fishel[1,†]

[1]Department of Cancer Biology and Genetics, Department of Surgery, and Division of Surgical Oncology, [2]Biomedical Sciences Graduate Program, The Ohio State University Wexner Medical Center, Columbus, Ohio, 43210

Keywords: single-molecule imaging, nanoscopy, force spectroscopy, particle tracking, 3D localization

* These authors contributed equally

† Correspondence should be addressed to: Richard Fishel (fishel.7@osu.edu)

Abstract

Magnetic tweezers are a popular biophysical instrument for manipulating and measuring single molecules. Most groups rely on custom-built setups tailored to specific experiments, making it challenging to implement and share software. Typically, image acquisition and hardware control are automated via LabVIEW, while real-time video processing is implemented in C++/CUDA libraries. Live processing can eliminate the need to store raw video, enabling high throughput, fast acquisition rates, and simplified experimental workflows. However, no open-source general-purpose software framework currently unifies these capabilities for magnetic tweezers experiments. Here, we introduce MagTrack and MagScope open-source Python-based tools designed to fill this gap. MagTrack is an image-processing library that efficiently determines bead-positions from magnetic-tweezers videos using CPU and/or GPU computation. MagScope is a comprehensive software framework offering a graphical user interface, real-time hardware control, data acquisition, and video processing. It is built on a multiprocessing architecture for responsive, high-throughput computation. Together, MagTrack and MagScope offer a fully customizable, end-to-end, open-source Python alternative to proprietary or fragmented systems, enabling laboratories to adapt and extend the framework according to their experimental needs.

Main

Single-molecule force spectroscopy has revolutionized the ability to measure and manipulate the mechanics and kinetics of individual biomolecules.[1-3] Among the various platforms available, magnetic tweezers have become popular due to their relative simplicity, low cost, and inherent high throughput.[4-14] Although these instruments can be built in a variety of configurations, their core concept remains simple: a microscope with a pair of magnets positioned above the sample **(Figure 1a)**. In a typical experiment, DNA, protein, or other polymers are tethered between a microscope slide and micron-sized magnetic beads. The magnets exert a tunable force and torque on the beads, controlled by moving the magnets. With a modest investment in a suitable camera, a capable computer, a few machined components, and standard optical parts, one can construct a high-resolution magnetic tweezers instrument capable of detecting nanometer-scale changes in molecular extension.[15,16] Beyond their high-resolution, magnetic tweezers inherently support parallel measurements, as multiple beads can be tracked simultaneously within a single camera field of view. This high-throughput capability makes magnetic tweezers particularly valuable for studying biological processes that exhibit stochastic behavior, such as enzymatic activity and protein folding.[16,17]

A challenging aspect of building a high-throughput magnetic tweezers system is the software required to process video recordings. Magnetic tweezers videos can be analyzed with generic standalone video-tracking tools such as ImageJ.[17-19] However, achieving high spatial and temporal resolution demands cameras that generate large images at high frame rates, resulting in enormous data volumes (1-100 GB/min). Handling such data creates a significant bottleneck in experimental workflows. Instead, many instruments are operated with software to process the video feed in real-time. This approach not only reduces data storage but also enables researchers to monitor and respond to the results.

The most common platform for live processing in magnetic tweezers experiments is LabVIEW, which is well-suited for hardware interfacing and instrument control. An influential

implementation from the Dekker group combined LabVIEW with a powerful custom-built C++/CUDA library for real-time image analysis.[20,21] However, LabVIEW is proprietary, expensive, and not widely used outside of instrumentation limiting its reach. Recent advances in generative AI-assisted coding have made text-based programming more accessible.[22] However, these benefits do not extend to LabVIEW, which uses a graphical programming environment. Other groups have implemented real-time magnetic tweezers software in C++ or Python, but these solutions are specific to individual laboratories and lack the documentation necessary for broad adoption.[4,12,16,20,21,23-25]

These challenges have highlighted a recognized need for a modern, open-source, and accessible software framework for magnetic tweezers.[6] Ideally such software should be written in Python, which is widely used in biophysics and has a rich ecosystem of scientific libraries. Indeed, many existing post-processing tools for magnetic tweezers data analysis, such as *Tweezepy*, are already Python-based.[26]

Here, we offer a fully Python-based framework for both image processing and live data acquisition. We introduce two complementary open-source projects: *MagTrack*, a GPU-accelerated image processing library for magnetic tweezers, and *MagScope*, a customizable interface for real-time instrument control and acquisition. Both projects were developed with thorough documentation to guide users through setup and designed to be customizable. We characterize the capabilities of this framework using real datasets and standard experiment workflows. Three key metrics of real-time magnetic tweezers software are evaluated: 1) stability, 2) throughput, and 3) resolution. The software and instrumentation were capable of collecting and processing data over multiple days with no breaks in recordings. At low magnification (10x) we tracked thousands of beads simultaneously. At high magnification (100x) we could accurately track beads with sub-nanometer resolution. This new framework lowers the barrier to access and customize magnetic tweezers software.

Results

Program Design and Development

Our goal was to design a framework for magnetic tweezers image processing and live acquisition with performance comparable to existing C++/CUDA approaches, yet more convenient for laboratories to adopt and modify. We therefore focused on high-level languages widely used in biophysics, particularly Python and MATLAB, which are already common for post-processing of magnetic tweezers data. A unified approach in one language would enable end-to-end data collection and analysis all in a high-level language, simplifying experiment workflows and increasing user accessibility.

Programs written in high-level languages are often viewed as slower than low-level counter parts even in raw numerical performance. While this is sometimes unavoidable, it is often possible to achieve similar results by using optimized libraries/routines. To assess feasibility, we implemented a center-of-mass function that converts image stacks into x-y positions in C, C++, Rust, MATLAB, and Python, and benchmarked a 100 × 100 × 100 stack over 10,000 iterations. Overall, the five implementations performed similarly with total runtimes ranging from 1.43 seconds to 3.30 seconds **(Figure 1b, left)**. The only language that stood out was Python, which was about twice as slow, a significant but acceptable decrease **(Figure 1b, left)**. Despite this, we measured comparable performance to lower-level implementations for compute-bound tasks utilizing Python and MATLAB that would be typical in magnetic tweezers image processing.

Python and MATLAB have demonstrated limitations in performance that includes hardware interactions, multi-threading/multi-processing, and graphical user interface (GUI) compared to lower-level languages. Therefore, we compared Python and MATLAB in a more realistic end-to-end setting with a GUI, live camera acquisition, and real-time x-y-z tracking. Under these conditions MATLAB processed only about one tenth as many images as Python **(Figure 1b, right)**. In addition, MATLAB, like LabVIEW, is proprietary while Python is free and open-

source. Given that lower-level languages such as C++/CUDA require more specialized expertise to modify, we chose to adopt Python as the basis for our control and analysis framework.

We organized the project into two Python packages. MagTrack is a library of image-processing routines for magnetic tweezers videos, while MagScope is a customizable multiprocessing application that provides a GUI, hardware interfaces, live acquisition, and calls to MagTrack for real-time analysis. This separation allows MagTrack functions to be used independent of a microscope, such as for offline video processing.

MagTrack for image processing

The MagTrack library covers a wide variety of standard magnetic-tweezers image-processing algorithms including: center-of-mass, auto-convolution, radial-profiling, and z-look up.[13,16,20] For users who want a straightforward solution we created a complete pipeline that converts images to x-y-z tracks **(Figure 1d)**. Users can also choose to mix and match components to build their preferred pipeline or integrate them with a function of their own design.

Many of the MagTrack functions have variations included for different use cases. For example, a bead's radial-profile can be determined directly from an image if its x-y center is known. This is a common approach for groups that use the radial-profile only for determining the z position.[12,23] A second case is to use the radial profile as part of the QI algorithm to refine the location of a center.[20] A third case is to use a Fourier transformed image to determine the radial-profile for z-look up without a known center. This fast approach is useful for groups that only need to determine the z position and not x-y.[27]

Details of each algorithm are included in the methods section as well as in the project's documentation (see online repository). Briefly, we implemented all the algorithms using NumPy[28], CuPy[29] and SciPy.[30] NumPy and SciPy are widely used in scientific computing, are well documented, and extensively tested. CuPy provides a NumPy-like interface to CUDA-enabled GPUs for faster computation. As a result, we only needed to write the image processing

algorithms once and they can be run on either a CPU or GPU with no additional code or knowledge of GPU programming **(Figure 1c)**. GPU acceleration only requires a CUDA-compatible GPU, otherwise, the program automatically falls back to CPU execution.

Given that the algorithms implemented in MagTrack are standard throughout magnetic tweezers field, we were not surprised to see the similar accuracy as previous implementations.[20] However, as discussed earlier our optimized Python implementations were slower than C++ on a CPU. Similarly, when comparing performance on a GPU with kernels written directly in CUDA compared to that generated with Python through CuPy, we observed slightly slower performance. However, these effects appeared to be relatively insignificant. To evaluate this more rigorously we developed a physics-based simulation to create artificial images of beads. The simulation can create images with beads at different image sizes, magnifications, noise levels, and x-y-z positions. We then created a system to automatically measure and log the runtime of every function included in MagTrack on both CPU and GPU. We then used this to evaluate nine different computers ranging from budget Windows laptops to MacBooks to scientific workstations. Logs of each run are included in the online repository along with scripts/documentation on how to conduct the benchmarking.

By far the largest effect on image processing runtimes is the size and region-of-interest (ROI) of the images as well as the number of images batched together in a stack to be simultaneously processed. The runtime in turn inversely affects the throughput (number of images processed in one second). An example of a few algorithms run on an Intel i7-13700 CPU is included in **Supporting Figure 1a**. We compare three different ROI sizes that cover the range of those commonly used: 16x16, 64x64, and 256x256. For faster algorithms like the z-look up, batches between 10 to 200 frames take the same amount of time to process despite being larger. Thus, larger batches up to 200 frames result in better performance. Beyond 200 frames, the runtime grows linearly, yielding no gain in performance for the larger batch size. Conversely, for slower algorithms such as generating the fast-Fourier transformed (FFT) radial profile, CPU

runtime shows a linear increase in CPU runtime with batch size. This relationship is more easily examined as throughput, which is defined as the number of images that can be processed in 1 second broken into batches **(Supporting Figure 1c)**.

We also evaluated performance on several GPUs. The results with an NVIDIA RTX 3070 are shown in **Supporting Figure 1b,d**. Because GPUs have a smaller dedicated memory but many more processors, they benefit substantially from processing image stacks in optimal batch sizes. The GPU often, but not always, outperformed the CPU. When MagTrack video processing runs on a GPU within MagScope, CPU time can be dedicated to the microscope's operation (see below). With advances in processor performance over the last decade the performance of individual algorithms is not as critical as it once was. Therefore, the somewhat slower performance of a Python implementation compared to lower languages implementation is not as critical. Overall, MagTrack is suitable for processing the throughput of most magnetic tweezers experiments.

MagScope for live microscope control, acquisition, and processing

MagScope was designed to replace the need to use LabVIEW. Besides avoiding this proprietary software, Python provides much more control over the way data is processed and moved across the program to create a fast optimal experience. We adopted a multiprocessing architecture in which each major module runs as a separate process **(Figure 1e)**, allowing CPU-intensive tasks to be distributed across cores. This design also avoids Python's global interpreter lock (GIL). We created an inter-process communication system that enables calls between processes. At program launch, each process is spawned as a child process of MagScope **(Figure 1e)**.

A primary requisite is that MagScope be flexible to work with any microscope hardware (cameras, motors, electro-magnets, lights, pumps, etc.) while still providing a clear path forward to allow hardware to communicate with the system. Our first consideration was whether to use an existing Python microscope-control framework such as Python-Microscope,[31] PyImageJ,[32] or

Microscope-Cockpit.[33] These frameworks provide robust hardware abstractions, but in our testing each limited camera frame rates below those required for fast magnetic tweezers acquisitions.

At the same time, many cameras and other hardware commonly employed with magnetic tweezers microscopes have excellent free Python libraries. After testing several different camera brands and types as well as motors, pumps, stages and light sources, we decided to leave hardware library selection up to the user. In our implementation, we rely primarily on manufacturer-supplied Python libraries, though equivalent functionality can be achieved via third-party packages.

Given the large volume of video data generated in typical magnetic tweezers experiments, it is important to avoid copying data between sub-processes. We designed a custom ring buffer to create a shared memory space for video and tracking data **(Figure 1f)**, allowing producer-consumer style communication in which acquisition processes write frames and analysis processes read them without repeated allocation. Ring-buffers ensure that space for these large arrays is allocated once for the whole program and data is only written or read by one process at a time.

MagScope comes with many standard features such as Z-LUT generation, live plotting, and a live feed of the camera's field of view. It also includes several powerful built-in features that include a lightweight scripting runtime that allows the user to queue up GUI interactions and hardware commands for experimental repetition. It also includes a XY-lock, which when enabled uses the tracking positions of the beads to update the ROI's positions so that each bead remains in the center of the ROI. This is particularly helpful for long duration experiments where drift can cause the beads to slowly exit the ROIs. Similarly, it includes Z-lock which when engaged, a single reference bead z position can be used to control a focusing piezo motor to keep the entire FOV within range of the Z-Look up table (Z-LUT). This is particularly useful for high-resolution and extended duration measurements.

By default, MagScope launches with a simulated camera, allowing users to explore the interface and test acquisition and analysis pipelines without specialized hardware **(Figure 1g)**. This is also critical during development for testing new features. We created abstract base classes for users to extend both general child processes and hardware-specific child processes. MagScope comes with an example of how we extended these classes for implementation by our group **(Figure 1e)**.

Availability, testing and documentation

Both MagTrack and MagScope are available on the Python Package Index (PyPI), are released under the GPL-3.0 license, and are thoroughly documented. This includes installation guides, configuration examples, API references, and example analysis notebooks. Both programs were developed with continuous integration to build and test on Windows, macOS and Linux operating systems, as well as multiple versions of Python and their dependencies. Source code, documentation and benchmarking runs are hosted on GitHub:

https://github.com/7jameslondon/MagTrack

https://github.com/7jameslondon/MagScope

High-resolution 3D bead tracking

The performance of the framework was evaluated on a physical microscope. We implemented hardware configuration files for the camera and motor stages, specifying device parameters and control commands to enable communication with MagScope. Tracking performance was characterized in a high-resolution configuration (100x magnification, 50 nm/pixel, 256×256 ROI). Polystyrene reference beads with a 3-μm-diameter were fixed on a microscope slide, and their positions were collected at 145 Hz with Z-Lock feedback enabled **(Figure 2a)**. The maximum resolution determined using the Allan deviation of the fixed bead's track was 0.29, 0.25 and 0.55 nm, in x, y and z, respectively **(Figure 2a)**. Subtraction of this trace from another, eliminated

instrument drift and low-frequency vibrations **(Figure 2b)**. This improved the maximum resolution to 0.04, 0.03 and 0.13 nm, in x, y and z, respectively **(Figure 2b)**.

Next, we explored the resolution using DNA-tethered beads. Magnetic beads with a 2.8-µm-diameter were tethered to a flow cell with a 4 kb DNA and kept at 1 pN of force. The tethered beads, and reference beads on the surface of the flowcell, were tracked in x, y and z at 100 Hz with the Z-Lock enabled. We subtracted the reference bead's position from the tethered bead's position which preserved the tethered bead's Brownian fluctuations while suppressing shared instrument drift, providing a realistic scale of experimental noise **(Figure 2c)**. The accompanying Allan-deviation analysis reflects higher variance due to inherent Brownian motion of the tethered bead at 1 pN **(Figure 2c)**. Both the reference bead and tethered bead measurements are in close agreement with the maximum resolution achieved by other groups with high-resolution configurations.[11] This validates the accuracy of the standard pipeline included in MagTrack with real beads.

Long-term tracking stability

We tested the operation of MagScope while collecting data continuously at 100 Hz for up to a week at a time. During all our testing we have never experienced a crash or other instability from the software. Several groups have created routines to maintain beads' relative x-y-z positions under instrument drift. As discussed previously, we have created a XY-Lock and Z-Lock feature that maintains the beads in the center of their ROIs and a reference bead's focus. This is particularly useful for both long-term and high-resolution measurements. To demonstrate the consistency of MagScope and the XY/Z-Lock we tracked reference beads for over 4 days without interruption (**Figure 2d**). The results underscore the robustness and practicality of MagScope in real-time acquisition for extended durations.

Force Control, Resolution, and DNA Conformational Kinetics

Having demonstrated the resolution and stability of our software framework, we next sought to determine how this accuracy can be combined with hardware control to make practical measurements of DNA kinetics. Hairpin rupture and unfolding have repeatedly been quantified with magnetic tweezers microscopy.[15] These sensitive measurements rely on the detection of subtle changes in DNA extension, fine force control, and millisecond-scale kinetics detection. We probed a 1 kb DNA containing a 60 nt palindromic hairpin with a nick in the opposite strand (**Figure 3a**).[34] We recorded the beads at 145 Hz in the same high resolution configuration as before. Using the built-in scripting feature of MagScope we performed a series of gentle force ramps (from 6 pN to 16 pN at 0.2 pN/s) and twists (20 turns, 0.1 turns/s, 5 pN) to identify single DNA tethers. During the final force ramps single-molecule traces resolved discrete 20 nm jumps in the DNA extension corresponding to hairpin rupture as the force was increased and hairpin re-formation as force was decreased (**Figures 3c**). Examining the force at which 18 molecules ruptured and reformed revealed a tight distribution in both directions centered around 13.7 pN (**Figure 3b**).

Similarly, we used MagScope to perform steps of force clamps starting at 12 pN up to 14 pN in 0.5 pN steps. Traces of the DNA's extension at 13 pN showed a fast equilibrium as the hairpin repeatedly opened and closed **(Figure 3d)**. Increasing or decreasing the clamp force resulted in a corresponding higher or lower occupancy in the ruptured state **(Figure 3d)**. The ramp data shows modest hysteresis, rupture exceeding re-formation expected for this finite rate of force ramp. The nick facilitates rapid re-hybridization without topological constraints, producing clean two-state switching. This assay demonstrates that our MagScope/MagTrack framework can be used in practical experiments to precisely control the instrument hardware and acquire high spatial and temporal resolution data.

High-Throughput Flow-Stretching

One of the largest advantages that magnetic tweezers offer over other force spectroscopy methods is its inherent multi-throughput by imaging/pulling multiple beads simultaneously. We examined the performance of MagScope in a high throughput analysis. To demonstrate the versatility of MagScope we performed magnetic tweezers flow-stretching experiments on a 4kb DNA in a high-throughput configuration (10x magnification, 500 nm/pixel, 28×28 ROI) (**Figure 4a**). We simultaneously tracked 1,000 1-μm-diameter beads at 100 Hz (**Figure 4b**). Under a moderate magnetic force of 1 pN non-specific interactions between the beads and surface was minimized (**Figure 4a**). A steady flow of buffer through the flowcell produced an increase in DNA extension that relaxed to the starting position when the flow was stopped (**Figure 4c**). Traces from molecules that appear to be stuck, tethered by multiple DNA molecules, or had broken tethers that failed to return to baseline were excluded with a series of quality control filters, producing 338 high quality traces (**Figures 4b,c**). The DNA extension increased with flow and plateaued near the 4 kb contour length (1.36 μm) at 200 μL/min (**Figures 4c,d**). Repeating the analysis at different flow rates showed that the extension reaches is maximum by 200 μL/min (**Figure 4e**). Overall, these observations demonstrate that MagScope/MagTrack can be utilized for high-throughput measurements as well as high-resolution analysis.

Discussion

In this work, we have developed and validated an accessible open-source Python framework for magnetic tweezers image processing and real-time instrument control. The MagTrack library provides CPU and GPU bead tracking algorithms. MagScope is a customizable package for live data acquisition and microscope control. We demonstrate the computational speed and tracking resolution of our approach and achieved a maximum spatial resolution of approximately 0.1 nm. The performance of the approach was further supported with a practical DNA hairpin extension analysis. We easily resolved 20 nm jumps in the DNAs extension. We also demonstrate the

performance of bead tracking and data acquisition in a high throughput experiment while simultaneously tracking 1,000 beads in a single field of view.

The new Python framework establishes a baseline for additional development and customization that can be integrated into specific experimental needs. We acknowledge that groups with existing proprietary or in-house software systems may need to perform additional integration to connect their hardware with Python, though the framework has been designed to integrate such adaptations as straightforward as possible. This framework provides a cost-free alternative to LabVIEW-based approaches while leveraging the flexibility and extensibility of the Python ecosystem. Importantly, the open-source nature of this platform enables seamless incorporation of emerging machine-learning-based video tracking algorithms, most of which are developed in Python.[35] Future work will focus on integrating deep-learning approaches to further enhance tracking performance and automation.

Methods

Source Code

Source code, documentation and stable releases of both MagTrack and MagScope are available online.

https://github.com/7jameslondon/MagTrack

https://github.com/7jameslondon/MagScope

Direct language comparison with center-of-mass image processing

To evaluate the practicality of using Python or MATLAB for image processing, we started by evaluating the center-of-mass algorithm in various languages **(Figure 1b, left)**. This is the most basic bead tracking algorithm and is used by most groups to at least get a coarse estimate of the x-y position, often before refining it with other algorithms. The center-of-mass algorithm is easily vectorized. In lower-level languages this is easily optimized by the compiler. In Python and

MATLAB, precompiled libraries achieve similar efficiency for the routines covered, such as taking sums over arrays. However, these efficiencies can be somewhat reduced when intermediate calculations are arranged outside of the optimized routines. Conversely, low-level languages usually require a deeper understanding of programming and produce a longer and more complex code. To determine how the code complexity would vary between languages and how the runtime of the code would compare, we wrote the center-of-mass routine in C, C++, Rust, MATLAB, and Python. We attempted to optimize the code as much as possible. Aggressive compiler optimizations ("-O3") and fast math standard ("-ffast-math") were used to compile C, C++ and Rust. We did not implement just-in-time (JIT) compilation in Python (such as with Numba). While this might improve the Python execution time it would also make the code much more complex. Because one of our overarching goals was to make the code relatively understandable, we chose not to implement these optimization protocols. The source code and instructions are included in the repository within the benchmarking directory. In each case we created a routine for the center-of-mass algorithm, and then a separate runner to measure the runtime. For each, a 100 × 100 × 100 stack of ones was created as a fake test case. Each routine was first executed for a warm-up and then the actual runtime was measured over 10,000 iterations.

Complete end-to-end comparison

For our end-to-end comparison between MATLAB and Python **(Figure 1b, right)**, we first created a straightforward image processing pipeline in form of stacks of images and returned profiles and x-y-z positions. This consisted of a center-of-mass routine to estimate the x-y positions, then a standard sub-pixel auto-convolution routine repeated five times to refine the x-y positions, then a radial profile estimated directly from the image stack with radial pixel binning, and finally a Z-Look-up against a pre-determined Z-LUT. These routines are the same as those specified in the final Python implementation in MagTrack. We then used manufacturer's libraries to connect the camera and three motors to MATLAB and Python. A basic GUI was created to control the motor

positions, basic camera settings such as framerate. Finally, this was all coordinated by a launcher that setup the GUI and passed instructions to the appropriate functions for the hardware or to process images. This launch made use of multi-processing wherever possible in both cases, and in both cases the image processing was performed on the GPU. To benchmark each program, we initiated them each separately and ran them while no other program was operating. Ten ROIs were selected and at each program report the average number of frames processed per second. The inverse was used to find the runtime per frame.

Image processing algorithms

Center-of-mass: This is a simple routine to obtain initial estimates of bead positions from square image stacks. The input image stack should be arranged as a three-dimensional array of size N×N×T, where N is the number of pixels and T is the number of frames. For each frame, the algorithm treats the pixel intensities as a mass distribution, and computes x and y centroid coordinates. Before calculating the centroids, each frame can optionally be background-corrected by subtracting either the frame-wise mean or median intensity and taking the absolute value, which reduces the influence of uniform background offsets. The corrected images then collapse along the x- and y-axes to obtain one-dimensional intensity profiles, and the first moment of these profiles is evaluated with respect to the discrete pixel coordinate grid. Dividing these moments by the total frame intensity yields the x and y positions for each frame.

Auto-convolution: This is used to refine bead center positions given an initial estimate. For each frame in the image stack, given x and y coordinates were used to select the corresponding central column and row. These one-dimensional intensity profiles are mean-subtracted and multiplied by a Gaussian window centered on the estimated position, which suppresses edge contributions and emphasizes the region near the bead. The algorithm then computed the auto-convolution of each profile using an FFT-based convolution and identified the location of the maximum in the

row and column convolutions. Because the auto-convolution peak is centered on the point of highest symmetry, these peak positions were converted back to image coordinates to obtain updated x and y bead positions for each frame. When needed for later sub-pixel fitting, the full convolution profiles can be retained instead of only the peak indices.

Sub-pixel auto-convolution: This feature is identical to auto-convolution. Once the convolution and maximum are found, a small, fixed-width neighborhood (typically 5 points) is extracted and fit with a quadratic function. The vertex of this fitted parabola provides a sub-pixel estimate of the convolution peak position along the row and column. These refined peak locations are then mapped back into the original image coordinate system, accounting for the symmetry of the auto-convolution, to yield sub-pixel x and y bead coordinates for each frame.

Auto-convolution multiline: This is the same as the auto-convolution but rather than one central row/column being evaluated, multiple rows/columns are averaged. The number of rows/columns can be provided by the user.

Sub-pixel auto-convolution multiline: The same as the auto-convolution multiline routine but with the sub-pixel fitting from the sub-pixel auto-convolution routine.

QI (Quadrant Interpolation): Generally, this is the same as the originally developed approached.[20] For each frame, the algorithm samples the intensity along the horizontal and vertical axes that pass through the current bead position, extracting short one-dimensional profiles centered on the estimated pixel. For each axis, three neighboring samples (center pixel and its immediate neighbors) are used to fit a quadratic function to the local intensity profile. The vertex of this fitted parabola provides a sub-pixel offset relative to the central pixel along that axis. These x- and y-offsets are then added to the input coordinates to yield refined bead positions for each frame.

Radial Profile: This is used to convert each bead image into a one-dimensional intensity profile as a function of distance from the bead center. For each frame in the image stack, the routine takes the corresponding x and y center coordinates and constructs a grid of pixel positions. It then computes the Euclidean distance from the center to every pixel and assigns each pixel to a radial bin based on this distance. An optional oversampling factor subdivides the native pixel-wide bins into finer radial intervals, increasing the resolution of the profile without changing the underlying image. Within each radial bin, the pixel intensities are averaged, yielding a smooth radial intensity profile for each frame.

FFT Radial Profile: For each frame in the stack, we computed a 2D real-valued only discrete Fourier transform and shifted the zero-frequency component to the center of the spectrum. The complex Fourier coefficients were converted to their magnitudes and then azimuthally averaged by grouping coefficients according to their radial distance from the origin in frequency space. To increase radial resolution without changing the underlying data, the radial coordinate was oversampled by an integer factor, and the magnitudes were accumulated into these oversampled radial bins. The resulting one-dimensional profiles were restricted to a user-defined range. Because this procedure operates directly on the Fourier magnitude, it does not require prior knowledge of the bead center.

Z-Look-up: We used a look-up-table-based routine to convert radial intensity profiles into axial (z) positions. For each frame, the bead image was first reduced to a one-dimensional radial profile as described above. These profiles were then compared to a precomputed Z-look-up table (Z-LUT) containing reference radial profiles acquired at known z positions. In the Z-LUT, the first row stores the calibrated z coordinates, and the remaining rows store the corresponding reference profiles (with the central pixel excluded from matching). For each measured profile, the routine

computes the Pearson correlation coefficient against every reference profile in the Z-LUT and identifies the reference index that yields the highest correlation. To obtain sub-plane accuracy, the correlation values in a small neighborhood around this maximum are fitted with a parabola, and the vertex of this fit is used to estimate a fractional index between LUT entries. This fractional index is then interpolated onto the calibrated z-axis stored in the first row of the Z-LUT, yielding a continuous estimate of the bead's z position for each frame.

Standard Pipeline: First the center-of-mass is called, the results are passed to auto-convolution, the results are then updated with sub-pixel auto-convolution multiline five times, then the radial profile is calculated, and finally the Z-look-up if a Z-LUT is provided.

XY-Lock and Z-Lock

The XY-Lock works by checking the most recent valid position of all currently selected beads either when the user requests it with a button push or when the user has enabled a timer. The timer can be set to any interval the user requests; we used ten seconds. The difference between the bead's positions and the center of the ROI is then calculated for both the x and y directions separately. Differences of less than one pixel for either direction are ignored. Larger differences are rounded to the nearest pixel and then ROIs are then moved. An upper limit can be specified by the user to prevent sudden large changes. Overall, this routine does not disrupt the bead's measured position because the positions are reported as the distance from the top left corner of the camera to the center of the bead. Thus, the position of a bead is the sum of the bead's position relative to the ROI and the ROI relative to the camera's field-of-view. If the ROI moves one pixel to the left, the bead will appear to move one pixel to the right relative to the ROI, so no change is observed. However, if the bead is too close to the edge of the ROI, a more noticeable change can be apparent due to the improved performance of the algorithms when the bead is near the center of the ROI.

The Z-Lock works a little differently. It can be triggered manually or on a timer as with the XY-Lock. But instead of updating all the beads it looks at the value of one bead (selected by the user). The user also provided a target z value and the difference from the target to the current value is found. This is then added to the objective/z focusing piezo motor's target value.

Microscope design

We built an inverted microscope following previously established setups[11,15,16,20]. An objective lens is mounted on a single-axis piezo actuator rigidly integrated into the microscope base, enabling precise motion along the optical (z) axis relative to a fixed flowcell. A separate optical rail post carries a light source and a linear actuator with pulley system. The linear actuator (Thorlabs X-LSQ075A-E01) translated a pulley system. One wheel of pulley is attached to a rotary motor and other to a 3D printed magnet holder that separates the magnet pair by 0.9 mm. The magnet holder was designed using computer-added design (CAD) software, and was 3D printed on a Stratasys J35 Pro using Vero White resin. The second wheel is coupled to a rotary stepper motor (Zaber X-NMS17-E01) to impose controlled rotations (rate and turn count) on the magnetic assembly via a belt. The entire vertical post is mounted on a XY-stage (Thorlabs TBB0606) to position the magnet pair in such a way that the light source, center of magnet pair and the objective lens share a common optical axis. The linear actuator translates the pulley/magnet assembly along the optical axis to vary magnet-flowcell separation, while the piezo adjusts objective-sample focus independently. MagScope coordinates the linear and rotary axes, executing programmed protocols for DNA extension and supercoiling. Images were captured using a CMOS camera (Vieworks VC-25MX2-M150I) and transferred to a CoaXPress frame-grabber (Euresys PC3622). The computer used for the magnetic tweezers experiments was equipped with 64GB of RAM, an Intel i5-12600KF CPU, and an NVIDIA GeForce RTX 3060 Ti GPU.

Flowcell assembly

Flowcells were assembled by sandwiching a 0.12 mm-thick double-sided tape spacer between two glass coverslips (each 0.17 mm thick). The original coverslips were each 24 × 50 mm. The bottom coverslip was used at its full size (24 × 50 mm), while the top one was trimmed to 24 × 40 mm. The double-sided tape was cut into a rectangle to define a channel of dimensions 0.2 x 42 mm and positioned between the coverslips in such a way that ~1 mm of channel length remained open at each end to serve as the inlet and outlet. A 0.28 mm inner diameter polyethylene tube (BD Intramedic, PE10) was used for the inlet whereas a 0.76 mm inner diameter polyethylene tube (BD Intramedic, PE60) was used for the outlet of the flowcell. The tubes were affixed at each end, using a short section of pipette tip as a coupling, to provide the inlet and outlet connections. The resulting flowcell volume was typically ~10 μL.

Experiment procedures

Magnetic tweezers experiments were carried out using a standard procedure based on previously established protocols that consist of the following steps. First the flowcell was rigidly mounted and the outlet was connected to a pump (kD Scientific LEGATO, 78-8110) with a 10 ml syringe. The flowcell was then washed with a 400 μl of purge buffer (10 mM Tris pH 7.5; 140 mM NaCl; 1 mM EDTA; 0.5% Tween-20). A solution of 25 μl of polystyrene beads (Duke Standards, 4K-03, 3 μm) with 75 μl TNE buffer (10 mM Tris pH 7.5; 140 mM NaCl; 1 mM EDTA) was vortexed and injected into the flowcell followed by two hours of incubation to non-specifically fix polystyrene beads onto the flowcell surface. A further wash with TNE buffer removed all the polystyrene beads that were not attached to the surface. These polystyrene beads served as fixed reference beads to correct for mechanical drift during the measurements. The flowcell surface was then treated with 100 μl of 0.1 mg/ml Digoxigenin antibody followed by 2 hours of incubation. Thereafter, the flowcell was passivated using 100 μl of BSA solution, TNEB buffer (10 mM Tris pH 7.5; 140 mM NaCl; 1 mM EDTA; 0.2 mg/ml BSA), with 30 minutes of incubation. A 100 μl

solution of 2 pM DNA, having digoxigenin at one end and biotin at the other, was prepared in TNEB and injected followed by 20 minutes of incubation. An additional wash with TNEB removed untethered DNA. Thereafter, 2 µl of M-280 streptavidin beads (ThermoFisher, 1120D) were washed twice and resuspended in 100 µl of TNEB and injected followed by a TNE buffer wash to get rid of beads that were not tethered to the free floating biotinylated end of tethered DNA. While 2.8 µm M-280 streptavidin beads were used to probe palindromic hairpin dynamics, a 1 µm Streptavidin C1 beads (ThermoFisher, 65001) were used in flow-stretching essay.

Additional purge of non-specifically attached magnetic beads was performed by applying 20 pN of force for M-280 beads and 5 pN of force for C1 beads, followed by a TNE wash. While measuring the kinetic response from palindromic hairpin DNA, the tethered DNA was subjected to a series of test ramps. For instance, 20 (+) turns and 20 (-) turns at 1 pN were used to ensure that extension of DNA remains unchanged during this scan. A single tethered DNA showed no change in DNA extension, whereas a multi-tethered DNA showed decrease in extension since the DNAs become entwined.

Force on the magnetic beads were measured using variance method, following a correction free approach.[36] Briefly, for stationary magnet positions, M-280 beads tethered to a 4kb DNA were tracked along with a nearby fixed bead at 145 Hz, and force was estimated as $F \approx k_B T z / \langle x^2 \rangle$. We confined the force calibration for 4 kb DNA to moderate-force regime to avoid high-force denaturation and pursue force range of interest between 1 to 20 pN.

Substrate construction

The 1 kb DNA containing a 60 nt palindromic hairpin was assembled from three parts. Two separate PCRs were performed on a template plasmid to create the two 500 bp arms. One of the two primers in one of the PCRs contained a 5' biotin and in the other PCR one primer contained a 5' digoxigenin. The non-biotin DNA end of each plasmid was digested with BsaI (NEB) to create two non- complementary overhangs. An oligo containing the hairpin section and the complement

to the two overhangs was then ligated to the two PCR products. This created one DNA containing a biotin on each end and the harpin in the middle.

The 4 kb DNA was created from three parts. A plasmid was digested with BsaI (NEB) to create the central 4 kb section with two non-complementary overhangs. Separately two PCRs were performed with the same plasmid as a template to create two 500 bp fragments. Biotin-dUTPs were added to the otherwise standard PCR mix of one of the PCRs. And digoxigenin-dUTPs were added to the other. Thus, one fragment was modified with multiple integrated biotins and the other with multiple digoxigenin. The two fragments were digested with BsaI (NEB) and ligated to the unmodified central 4 kb section creating a DNA with a total length of 5 kb with 500 bp of one end that is randomly interspersed on one end with biotin and the other end with digoxigenin.

ASSOCIATED CONTENT

Supporting Information: CPU/ GPU runtimes and throughput of MagTrack routines (DOC)

AUTHOR INFORMATION

Corresponding Author

Richard Fishel (fishel.7@osu.edu)

Author Contributions

J.L. and A.S. contributed equally. J.L. wrote the Python and MATLAB code. A.S., J.L, N.T., and T.S. built the magnetic tweezers instrument, performed magnetic tweezers experiments, prepared DNA substrates, and prepared flowcells. J.L., A.S., and N.T. developed the magnetic tweezers experiment protocols. Z.R. ported Python, MATLAB and ImageJ code. The manuscript was written by J.L., A.S., T.S., and R.F. All authors approved the final manuscript.

Funding Sources

This work was supported in part by National Institutes of Health grant CA067007, AI150496 and GM150003. T.S. is a recipient of The Huntington's Disease Society of America Donald A. King Fellowship 1485211.

ACKNOWLEDGMENTS

We would like to thank Michael Root of Thomas Jefferson University and Ross Larue of The Ohio State University Wexner Medical Center for their insightful comments and stimulating discussions throughout the preparation of this work. We would also like to thank Kathryn Glowinski, Daniel Levine and Kiran Bangalore for their advice and review of the programs' code.

REFERENCES

1 Ritort, F. Single-molecule experiments in biological physics: methods and applications. *Journal of Physics: Condensed Matter* **18**, R531 (2006). https://doi.org/10.1088/0953-8984/18/32/R01

2 Neuman, K. C. & Nagy, A. Single-molecule force spectroscopy: optical tweezers, magnetic tweezers and atomic force microscopy. *Nature Methods* **5**, 491-505 (2008). https://doi.org/10.1038/nmeth.1218

3 Woodside, M. T. & Block, S. M. Reconstructing Folding Energy Landscapes by Single-Molecule Force Spectroscopy. *Annual Review of Biophysics* **43**, 19-39 (2014). https://doi.org/https://doi.org/10.1146/annurev-biophys-051013-022754

4 Gosse, C. & Croquette, V. Magnetic tweezers: micromanipulation and force measurement at the molecular level. *Biophys J* **82**, 3314-3329 (2002). https://doi.org/10.1016/s0006-3495(02)75672-5

5 De Vlaminck, I. & Dekker, C. Recent advances in magnetic tweezers. *Annu Rev Biophys* **41**, 453-472 (2012). https://doi.org/10.1146/annurev-biophys-122311-100544

6 Dulin, D. in *Single Molecule Analysis : Methods and Protocols* (eds Iddo Heller, David Dulin, & Erwin J. G. Peterman) 375-401 (Springer US, 2024).

7 Strick, T. R., Allemand, J.-F., Bensimon, D., Bensimon, A. & Croquette, V. The Elasticity of a Single Supercoiled DNA Molecule. *Science* **271**, 1835-1837 (1996). https://doi.org/doi:10.1126/science.271.5257.1835

8 Sarkar, R. & Rybenkov, V. V. A Guide to Magnetic Tweezers and Their Applications. *Frontiers in Physics* **Volume 4 - 2016** (2016). https://doi.org/10.3389/fphy.2016.00048

9 Strick, T., Allemand, J.-F., Croquette, V. & Bensimon, D. Twisting and stretching single DNA molecules. *Progress in Biophysics and Molecular Biology* **74**, 115-140 (2000). https://doi.org/https://doi.org/10.1016/S0079-6107(00)00018-3

10 Vilfan, I. D., Lipfert, J., Koster, D. A., Lemay, S. G. & Dekker, N. H. in *Handbook of Single-Molecule Biophysics* (eds Peter Hinterdorfer & Antoine Oijen) 371-395 (Springer US, 2009).

11 Choi, H.-K., Kim, H. G., Shon, M. J. & Yoon, T.-Y. High-Resolution Single-Molecule Magnetic Tweezers. *Annual Review of Biochemistry* **91**, 33-59 (2022). https://doi.org/https://doi.org/10.1146/annurev-biochem-032620-104637

12 Huhle, A. *et al.* Camera-based three-dimensional real-time particle tracking at kHz rates and Ångström accuracy. *Nature Communications* **6**, 5885 (2015). https://doi.org/10.1038/ncomms6885

13 Croquette, V., Orero, J. V., Rieu, M. & Allemand, J.-F. in *Methods in Enzymology* Vol. 694 (eds Min Ju Shon & Tae-Young Yoon) 1-49 (Academic Press, 2024).

14 Ribeck, N. & Saleh, O. A. Multiplexed single-molecule measurements with magnetic tweezers. *Review of Scientific Instruments* **79** (2008). https://doi.org/10.1063/1.2981687

15 Dulin, D. *et al.* High Spatiotemporal-Resolution Magnetic Tweezers: Calibration and Applications for DNA Dynamics. *Biophys J* **109**, 2113-2125 (2015). https://doi.org/10.1016/j.bpj.2015.10.018

16 Tapia-Rojo, R., Mora, M. & Garcia-Manyes, S. Single-molecule magnetic tweezers to probe the equilibrium dynamics of individual proteins at physiologically relevant forces and timescales. *Nat Protoc* **19**, 1779-1806 (2024). https://doi.org/10.1038/s41596-024-00965-5

17 Agarwal, R. & Duderstadt, K. E. Multiplex flow magnetic tweezers reveal rare enzymatic events with single molecule precision. *Nature Communications* **11**, 4714 (2020). https://doi.org/10.1038/s41467-020-18456-y

18 Rueden, C. T. *et al.* ImageJ2: ImageJ for the next generation of scientific image data. *BMC Bioinformatics* **18**, 529 (2017). https://doi.org/10.1186/s12859-017-1934-z

19 Video Spot Tracker (software and manual) (2015).

20 van Loenhout, Marijn T. J., Kerssemakers, J. W. J., De Vlaminck, I. & Dekker, C. Non-Bias-Limited Tracking of Spherical Particles, Enabling Nanometer Resolution at Low Magnification. *Biophysical Journal* **102**, 2362-2371 (2012). https://doi.org/https://doi.org/10.1016/j.bpj.2012.03.073

21 Cnossen, J. P., Dulin, D. & Dekker, N. H. An optimized software framework for real-time, high-throughput tracking of spherical beads. *Review of Scientific Instruments* **85**, 103712 (2014). https://doi.org/10.1063/1.4898178

22 Peng, S., Kalliamvakou, E., Cihon, P. & Demirer, M. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590* (2023).

23 Lansdorp, B. M., Tabrizi, S. J., Dittmore, A. & Saleh, O. A. A high-speed magnetic tweezer beyond 10,000 frames per second. *Review of Scientific Instruments* **84**, 044301 (2013). https://doi.org/10.1063/1.4802678

24 Daldrop, P., Brutzer, H., Huhle, A., Kauert, Dominik J. & Seidel, R. Extending the Range for Force Calibration in Magnetic Tweezers. *Biophysical Journal* **108**, 2550-2561 (2015). https://doi.org/https://doi.org/10.1016/j.bpj.2015.04.011

25 Saha, P. *et al.* Exploring force-driven stochastic folding dynamics in mechano-responsive proteins and implications in phenotypic variation. *Nature Communications* **16**, 1018 (2025). https://doi.org/10.1038/s41467-025-55946-3

26 Morgan, I. L. & Saleh, O. A. Tweezepy: A Python package for calibrating forces in single-molecule video-tracking experiments. *PLoS One* **16**, e0262028 (2021). https://doi.org/10.1371/journal.pone.0262028

27 Popa, I. *et al.* A HaloTag Anchored Ruler for Week-Long Studies of Protein Dynamics. *Journal of the American Chemical Society* **138**, 10546-10553 (2016). https://doi.org/10.1021/jacs.6b05429

28 Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585**, 357-362 (2020). https://doi.org/10.1038/s41586-020-2649-2

29 Nishino, R. & Loomis, S. H. C. Cupy: A numpy-compatible library for nvidia gpu calculations. *31st confernce on neural information processing systems* **151** (2017).

30 Virtanen, P. *et al.* SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods* **17**, 261-272 (2020). https://doi.org/10.1038/s41592-019-0686-2

31 Susano Pinto, D. M. *et al.* Python-Microscope - a new open-source Python library for the control of microscopes. *J Cell Sci* **134** (2021). https://doi.org/10.1242/jcs.258955

32 Rueden, C. T. *et al.* PyImageJ: A library for integrating ImageJ and Python. *Nature Methods* **19**, 1326-1327 (2022). https://doi.org/10.1038/s41592-022-01655-4

33 Phillips, M. A. *et al.* Microscope-Cockpit: Python-based bespoke microscopy for bio-medical science. *Wellcome Open Res* **6**, 76 (2021). https://doi.org/10.12688/wellcomeopenres.16610.2

34 Shon, M. J., Rah, S.-H. & Yoon, T.-Y. Submicrometer elasticity of double-stranded DNA revealed by precision force-extension measurements with magnetic tweezers. *Science Advances* **5**, eaav1697 (2019). https://doi.org/doi:10.1126/sciadv.aav1697

35 Pineda, J. *et al.* Geometric deep learning reveals the spatiotemporal features of microscopic motion. *Nature Machine Intelligence* **5**, 71-82 (2023). https://doi.org/10.1038/s42256-022-00595-0

36 Ostrofet, E., Papini, F. S. & Dulin, D. Correction-free force calibration for magnetic tweezers experiments. *Scientific Reports* **8**, 15920 (2018). https://doi.org/10.1038/s41598-018-34360-4

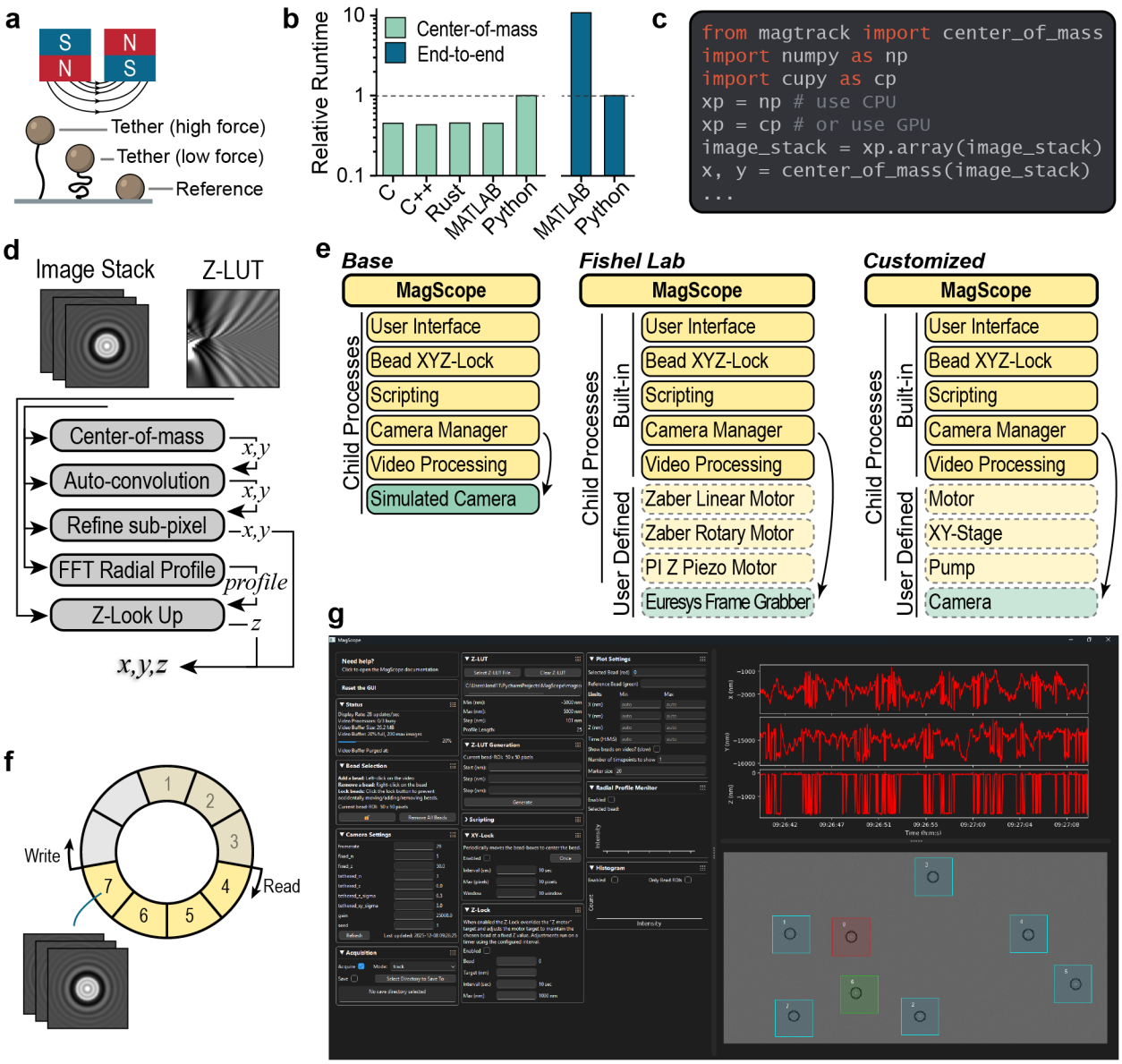

Figure 1. Overview of MagTrack and MagScope. **a)** Cartoon overview of magnetic tweezers principle. **b)** Relative runtime of center of mass algorithm in different programming languages (left). Relative runtime of end-to-end code in MATLAB and Python. **c)** Example of processor agnostic programing with MagTrack for a standard center-of-mass calculation on a CPU or GPU. **d)** Overview of tracking pipelines that can be composed with MagTrack. **e)** Three different use cases of the MagScope framework. First, the "Base" version which requires no additional setup to run. This is used for demonstration purposes and development testing. Second, the "Fishel Lab" implementation which was used by us to connect our specific hardware and preferred

controls to MagScope. Third, a “Customized” version to demonstrate the versatility of the MagScope framework for a laboratory-specific implementation. **f)** Diagram of how ring-buffers work to provide efficient inter-process access to large datasets including the video stream. **g)** Screenshot of MagScope base user interface.

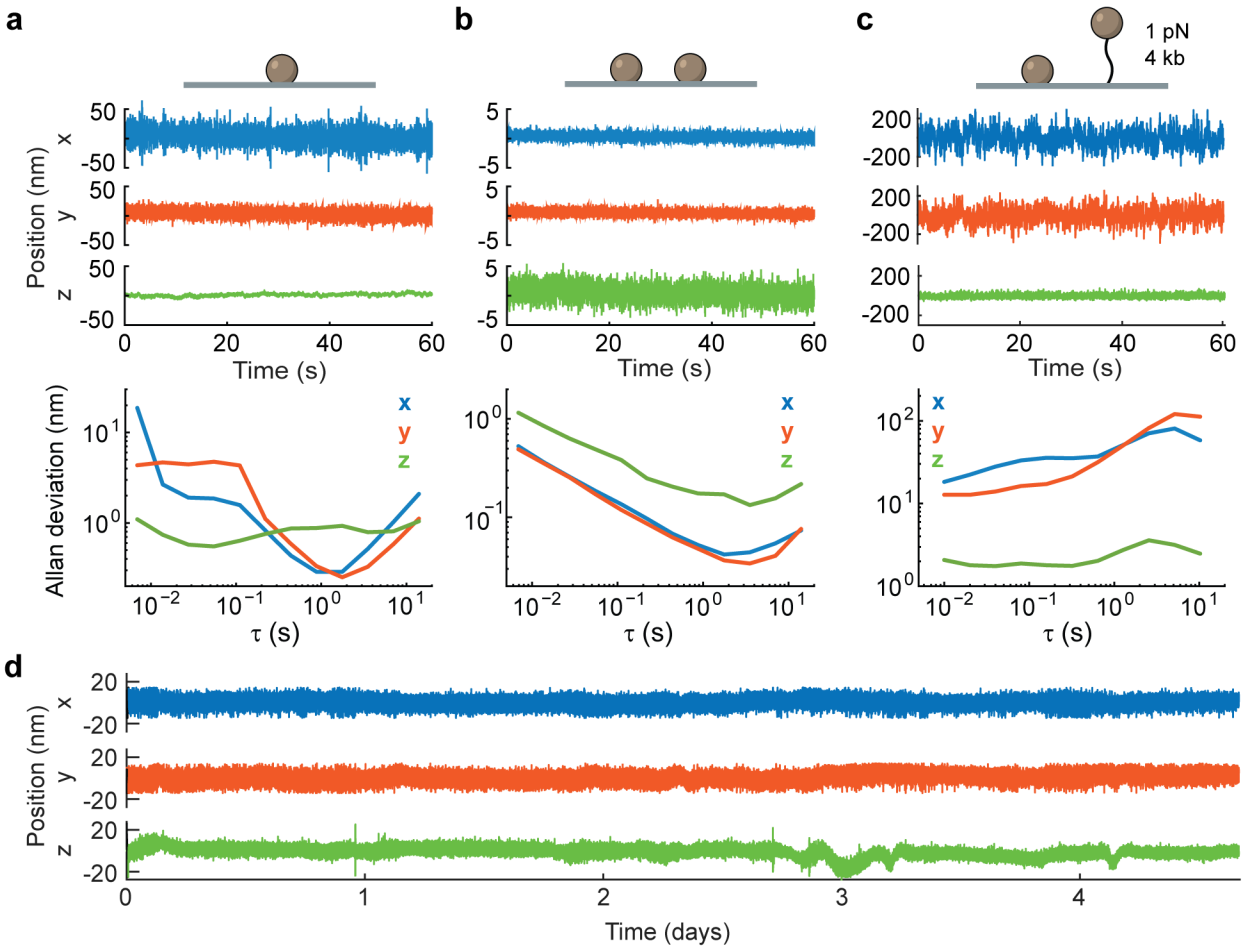

Figure 2. High-resolution and extended duration 3D tracking with MagTrack and MagScope. a) Raw data from a fixed bead collected at 145 Hz. (Top) Traces in x, y, and z. (Bottom) Corresponding Allan deviations from above. **b)** Same as panel a but after subtracting the positions of another fixed bead. **c)** Same as panel a but after subtracting the positions of a DNA tethered magnetic bead held at 1 pN of force and collected at 100 Hz. **d)** Traces of a fixed bead subtracted by another fixed bead in x, y, and z collected from an extended recording for more than four days at 100 Hz.

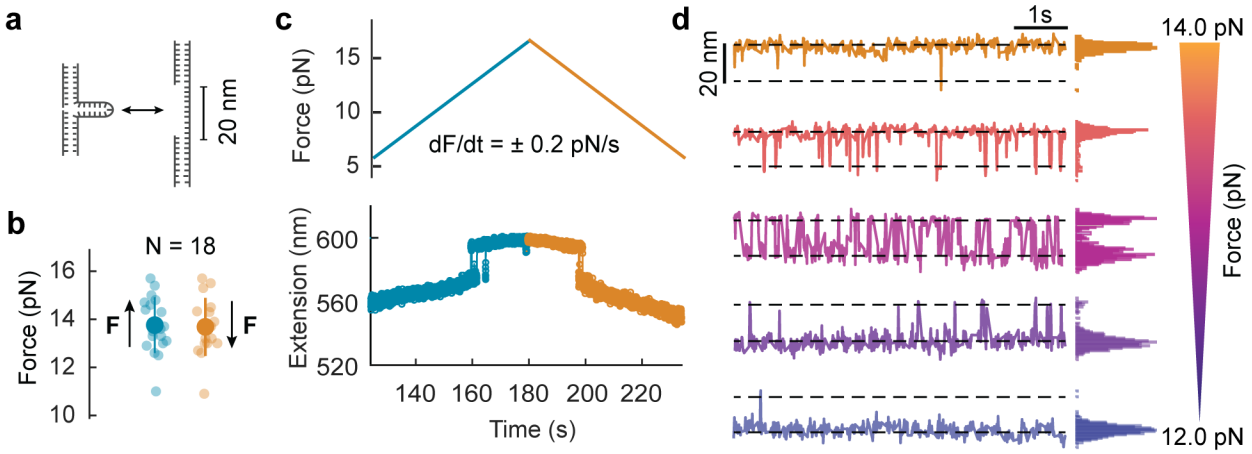

Figure 3. Dynamics of DNA hairpin folding. a) Schematic of a reversible two-state DNA hairpin. **b)** Distribution of rupture and reformation force of 18 molecules with mean (opaque circle) and standard deviation (error bar) are shown. Rupture force registered while increasing force ramp (F↑) and reformation force was registered while decreasing force ramp (F↓). **c)** Increasing and decreasing force ramp from 6 pN to 16 pN at 0.2 pN/s (top panel) and its response on the nicked hairpin DNA is shown by a representative trace where hairpin ruptures in force ramp up (F↑) at ~160 s and reforms in force ramp down (F↓) at ~200 s. **d)** Two-state kinetics of nicked hairpin DNA at discrete equilibrium forces from 12 pN to 14 pN at steps of 0.5 pN is shown with the occupancy histograms with kernel density estimates. The population shifts from predominantly open to predominantly closed hairpin state with decreasing force, with an even occupancy near 13 pN.

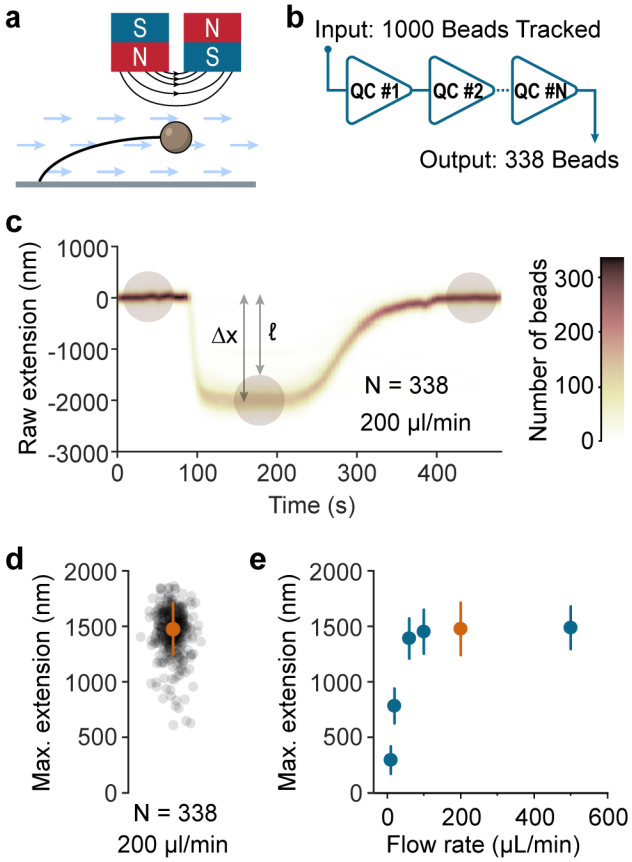

Figure 4. Flow stretching 1,000 molecules. **a)** Assay schematic: 4 kb DNA tethered to glass surface at one end and to a 1 µm magnetic bead at the other, a moderate magnetic force of 1 pN persists throughout the assay while flow applied the hydrodynamic drag. **b)** Schematics of the data processing scheme. 1000 beads were simultaneously tracked, while registering their x, y and z positions. These tracks were then processed through a series of quality control filters to eliminate beads showing atypical trajectories. **c)** The 2D kernel-density of real-time raw bead positions (via bead-center displacement) during a step of steady flow of 200 µL/min, followed by subsequent relaxation upon return to zero flow is shown. The kernel-density was mapped for the accepted 338 molecules, selected from 1,000 initially tracked (10x objective; 28×28-pixel ROIs; 100 Hz). **d)** Distribution of maximum DNA extension, ℓ, for the QC-accepted subset (N = 338) after correcting raw extensions, Δx, shown in Panel c by subtracting the bead radius (500 nm). Marker shows mean and standard deviation. **e)** Maximum DNA extension at different flow rates

is shown. Extension approaches the 4 kb contour length (~1.37 μm) by ~200 μL/min with no further extension observed at higher rates.