A Python framework for magnetic tweezers real-time image processing and microscope control

James A. London*, Abhishek K. Singh*, Teague C. Svendsen, Naciye Esma Tirtom, Zachary A. Root, Richard Fishel

Department of Cancer Biology and Genetics, The Ohio State University Wexner Medical Center, Columbus, Ohio, USA

*These authors contributed equally

Abstract

Magnetic tweezers are a popular biophysical instrument for manipulating and measuring single molecules. Most laboratories rely on custom-built setups tailored to specific experiments, resulting in a variety of hardware and software implementations. Typically, image acquisition and hardware control are automated via LabVIEW and specialized C/C++/CUDA libraries for real-time video processing. Live processing eliminates 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 flexible, modern, and fully customizable open-source alternative to proprietary or fragmented systems, enabling laboratories to adapt and extend the framework according to their experimental and programming needs.

1. Introduction

Single-molecule force spectroscopy has transformed our ability to measure and manipulate the mechanics and kinetics of individual biomolecules[1]. Among these platforms, magnetic tweezers are popular because they are simple to build, inexpensive, and high throughput[2-4]. Although instruments can be built in a variety of configurations, the core concept remains simple: a microscope with magnets positioned above the sample. In a typical experiment, DNA, protein, or other polymers are tethered between a microscope slide and micron-sized magnetic beads; magnets above the sample exert force and torque on the beads, which can be tuned by moving the magnets. Additionally, 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[5,6]. Moreover, magnetic tweezers inherently support parallel measurements, since many beads can be observed 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[6,7].

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[7,8]. However, achieving a high spatial and temporal resolution demands cameras that generate large images at high frame rates, resulting in enormous data volumes, often gigabytes per minute. 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 requirements but also enables researchers to monitor and respond to the results in real-time.

The most widely used framework for live processing of magnetic tweezers data is LabVIEW, which is well-suited for hardware interfacing and instrument control. However, LabVIEW is proprietary, expensive, and not widely used outside of instrumentation, limiting its reach. Other groups have implemented real-time magnetic tweezers software in C++ or Python, but these solutions are specific to individual laboratories and not designed for adaptation [2,6,9-12]. A particularly influential implementation used LabVIEW combined with a custom-built C++/CUDA library for real-time image analysis[13,14]. This is a very powerful approach but is difficult to extend or customize, demanding adequate experience in programming. Recognizing these challenges, researchers have highlighted the need for a modern, open-source, and widely accessible software framework for magnetic tweezers[4]. Ideally this software should be written in Python due to its wide use in biophysics and rich ecosystem of scientific libraries. Indeed, many existing post-processing tools for magnetic tweezers data analysis, such as *Tweezepy*, are already Python-based[15].

Here, we present a fully Python-based framework for both image processing and real-time 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. We demonstrate three key metrics of real-time magnetic tweezers software: stability, throughput, and resolution. We were able to collect and process data over multiple days with no breaks in recordings. At low magnification (10x) we could track thousands of beads simultaneously. At high magnification

(100x) we could accurately track beads in x, y, and z with nanometer-scale resolution. This new framework lowers the barrier to access and customize magnetic tweezers software.

2. Program Design and Development

2.1 Overall design and development

Our overall goal was to design a more accessible framework for magnetic tweezers image processing and live acquisition while maintaining the speed and accuracy established by existing approaches. We wanted to build this in a high-level programming language that would be approachable for other researchers. The two languages that stood out to us to balance these objectives were MATLAB and Python. Both are widely used in the scientific community and are popular for post-processing analysis of magnetic tweezers tracking data[4]. MATLAB has several advantages over Python for this purpose, including built-in tools for interfacing with hardware and native tools for creating a graphical user interface (GUI). However, MATLAB is proprietary and subject to a license fee while Python is open-source and free. More importantly, despite spending significant time optimizing a MATLAB framework, we found Python was often faster for live image processing.

We split the project into two libraries. MagTrack is a library of functions for magnetic tweezers image processing. MagScope is a customizable multiprocessing application to run magnetic tweezers microscopes. It provides a GUI, hardware interface, live video acquisition, and calls on MagTrack for image processing. This separation allows MagTrack functions to be used independent of a microscope, such as for offline video processing.

For both libraries, we developed clear documentation to assist in their implementation and customization. They are easily installed using the Python Package Index (PyPI). The full source code, installation instructions, and links to documentation are available through GitHub: MagTrack at (https://github.com/7jameslondon/MagTrack/) and MagScope at (https://github.com/7jameslondon/MagScope/). Both projects are released under a free software license GPL-3.0.

2.2 MagScope design

Our first consideration with MagScope was whether to use an existing Python microscope library such as Python-Microscope[16], PyImageJ[17], or Microscope-Cockpit[18]. These are excellent systems for interacting with hardware. However, each library limited camera frame rates more than what was expected for fast magnetic tweezers acquisitions. With none of these projects could we stream and process kHz frame rates (1280x1280 8-bit frames). Furthermore, many cameras and other hardware commonly used in 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. When implementing our own hardware, we chose to use the manufacturers' Python libraries, although the same results can be achieved using a third-party library.

For the entire program we developed a simple multiprocessing implementation such that each module of the program runs as a separate process **(Figure 1B)**. This avoids Python's global interpreter lock (GIL). We created an inter-process communication system that enables calls between processes with minimal knowledge of inter-process programming. At program start, each process is spawned as a child process of MagScope **(Figure 1B)**. Given the 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 1C)**. 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. By default, MagScope launches with a simulated camera and is fully prepared for users to try its features. 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 our own implementation in our laboratory **(Figure 1B)**.

2.3 MagTrack design

We chose to write a native Python implementation of various standard image processing algorithms used in magnetic tweezers rather than creating a wrapper around existing C/C++/CUDA programs[3,6]. This gave us more control over the flow of data through the program when interfacing with the CPU, RAM and GPU. It also makes the underlying code more accessible to those who are familiar with Python but not C/C++/CUDA. We chose to only use NumPy[19], CuPy[20] and SciPy[21] to implement these algorithms. NumPy and SciPy are standard throughout scientific programming and have excellent documentation. They are well tested and highly reliable. CuPy[20] provides an interface to GPUs with a syntax that matches NumPy and SciPy. We chose this over other Python GPU libraries because its API closely mirrors these libraries. As a result, we only needed to write our image processing algorithms once and they can be run on either a CPU or GPU with no additional code or knowledge of GPU programming other than NumPy/SciPy **(Figure 2A)**. The only requirement for GPU acceleration is a CUDA-compatible GPU (commonly available on NVIDIA hardware). If a GPU is not available or CuPy is not installed, the program automatically detects this and defaults back to the CPU.

We provide a standard pipeline that combines several of these operations in the library **(Figure 2B)**. Different groups prefer different tracking algorithms. For example, some directly calculate the radial profile while others use a Fourier-transformed image[6,13]. To provide flexibility, we implemented several widely used algorithms, including the popular QI algorithm[13]. Additionally, we provide examples of how to use each function in an interactive Python notebook. Using these methods users can mix and match components to build their preferred pipeline or combine them with a function of their own design.

2.4 Video processing runtime and throughput

To evaluate the performance of MagTrack, we first developed a simulation to create artificial images given a pixel size and the beads' desired x, y, z positions. We tested this for accuracy and speed. Given that the algorithms implemented here are standard throughout magnetic tweezers field, we were unsurprised to see similar accuracy to existing algorithms[6,11,14]. However, given that we are operating in Python rather than directly in C/C++/CUDA, we were more concerned with the speed of the algorithms. We therefore developed and included in the MagTrack repository a suite of benchmarks for each subroutine. Batching images into stacks often leads to improved performance over processing each image one at a time because the same operations can be performed to each image in the stack simultaneously. To evaluate this, we tested the runtime of the algorithms on several CPUs. A representative example of a few algorithms run on an Intel i7-13700 CPU is shown in **Figure 2C**. Because runtime is closely tied to the width and height of the images, we tested 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 10-200 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 FTT 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 **(Figure 2E).** We also evaluated performance on several GPUs. The results with an NVIDIA RTX 3070 are shown in **Figure 2D,F.** Because GPUs have smaller per-core memory but many more processors, they benefit substantially from processing image stacks at optimal batch sizes. Although the GPU did not always outperform the CPU, it often did. When MagTrack video processing runs on a GPU within MagScope, CPU time can be dedicated to the microscope's operation. Overall, these algorithms are slightly slower than C/C++/CUDA counterparts. With advances in processor performance since the C/C++/CUDA algorithms were developed, these differences matter less. Overall, MagTrack is suitable for processing the throughput of most magnetic tweezers experiments.

3. Experimental Validation

3.1. 3D bead tracking resolution and stability

Next, we evaluated the performance of MagScope using MagTrack for video processing. We first examined the maximum resolution by tracking fixed reference beads at 100 Hz with a region-of-interest around each bead of 256x256 pixels **(Figure 3A).** Analysis of the track's Allan deviation shows a maximum resolution of ~1nm **(Figure 3A).** A corresponding trace obtained by subtracting one trace from another eliminated instrumental drift and low-frequency vibrations **(Figure 3B)**. This produced a maximum resolution of ~0.1nm. As a practical example we next examined a DNA tethered bead subtracted from a fixed bead kept at 1pN of force **(Figure 3C)**. This subtraction preserved the tethered bead's Brownian fluctuations while suppressing shared instrument drift, providing a realistic scale of experimental noise. The accompanying Allan-deviation analysis reflects higher variance due to inherent Brownian motion. The subtracted fixed bead demonstrates the instrument's precision, whereas the traces of a reference subtracted tethered bead demonstrate the practical precision during single-molecule measurements.

We have 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. To demonstrate the consistency of MagScope's performance we have included traces of fixed beads subtracted fixed beads over 1 min, 1 hour, and 14 hours **(Figure 3D)**. This proves the efficiency and robustness of the MagScope in real-time acquisition over extended durations without interruption.

3.2. Force Control, Resolution, and Conformational Kinetics

Next, we tested the resolution of MagScope/MagTrack in a practical assay. Hairpin rupture and unfolding have repeatedly been quantified with magnetic tweezers microscopy[5]. These sensitive measurements rely on the detection of subtle changes in DNA extension, fine force control, and millisecond-scale kinetics detection. We probed a 1kbp DNA containing a 60nt palindromic hairpin with a nick in the opposite strand **(Figure 4A)**.[22] We recorded ~7 beads(3 μm diameter) at a time a 100 Hz with 256x256 pixel ROIs and an image resolution of 50nm/pixel. Using the built-in scripting feature of MagScope we performed a series of gentle force ramps and twists to identify single DNA tethers. The script performed a force ramp from 6pN to 16pN and back to 6pN at a rate of 0.2pN/s. During the final force ramps single-molecule traces resolved discrete ~20nm jumps in the DNA extension corresponding to hairpin rupture as the force was increased and hairpin re-formation as force was decreased **(Figure 4B,C).** Examining the force at which 18 molecules ruptured and reformed revealed a tight distribution centered around ~13pN **(Figure 4D)**.

Similarly, we used MagScope to perform steps of force clamps starting at 12pN up to 14.5pN in 0.5pN steps. Traces of the DNA's extension at 13pN showed a fast equilibrium as the hairpin repeatedly opened and closed **(Figure 4E)**. Increasing or decreasing the clamp force resulted in a respectively higher or lower occupancy in the ruptured state **(Figure 4E)**.

The ramp data shows modest hysteresis, rupture exceeding re-formation expected for this finite rate of force ramp. The presence of the nick facilitates rapid re-hybridization without topological constraints, producing clean two-state switching. This assay demonstrates that MagScope/MagTrack can be used in practical experiments to precisely control the instrument hardware and acquire high spatial and temporal resolution data.

3.3. High-Throughput Flow-Stretching

Having tested MagScope/MagTrack for high resolution experiments, we next examined a high throughput experiment. To show the versatility of MagScope we performed magnetic tweezers flow-stretching experiments on a 4kbp DNA **(Figure 5A)**. We recorded 1,000 beads (1µm diameter) at 100 Hz with 28x28 pixel ROIs and an image resolution of 500 nm/pixel. A moderate magnetic force was used (~1 pN) to minimize non-specific interaction of beads from surface. A steady flow of buffer through the flowcell produces an increase in DNA extension that relaxes back when the flow is stopped **(Figure 5B)**. Traces from molecules that appear to be stuck bead, multi-tethered, and broken tethers that failed to return to baseline were excluded, producing 375 high quality traces **(Figure 5B).** The DNA extension increased with flow and plateaued near the 4 kb contour length (~1.3-1.4 µm) at ~200 $\mu L min^{-1}$ **(Figure 5B,C)**. Repeating this measurement at various flow rates showed no further increase at higher flow rates **(Figure 5D)**. Overall, this experiment demonstrates the MagScope/MagTrack are just as applicable for high-throughput measurements as they are for high-resolution.

4. Conclusion

In this work, we have developed a new open-source Python framework for image processing and real-time instrument control for magnetic tweezers. 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. With a maximum resolution of ~0.1nm. Furthermore, we tested the resolution of tracking with a practical DNA hairpin extension experiment. We were easily able to resolve 20nm jumps in the DNAs extension. We also evaluate the performance in high throughput experiments simultaneously tracking 1,000 beads at a time.

We believe this new framework establishes a baseline for future development. However, we acknowledge that this may not appeal to laboratories that have already invested resources into their own software. Additionally, it may take some effort to connect laboratory-specific hardware with Python. On the other hand, we provide a free alternative to LabVIEW-based approaches. Furthermore, developing this framework in Python opens the opportunity for further development in tracking algorithms. Many new approaches to video processing are emerging from advances in machine learning [23]. These approaches are often developed in Python. In future work deep-learning-based approaches could be integrated with this new framework.

5. Methods

Source Code

Source code, documentation and stable releases of both MagTrack and MagScope are available online. MagTrack at https://github.com/7jameslondon/MagTrack/ and MagScope at https://github.com/7jameslondon/MagScope through GitHub.

Microscope design

We built an inverted microscope following previously established setups. 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 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.

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 coverslip 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 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

The 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; 1mM 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; 1mM EDTA) was vortexed and injected into the flowcell followed by two hours of incubation to non-specifically fix polystyrene beads on the surface of flowcell. 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 while doing the measurements. The flowcell surface was then treated with a 100 µl of 0.1 mg/ml Digoxigenin antibody followed by 2 hours of incubation. Thereafter, flowcell was passivated using a 100 µl of BSA solution, TNEB buffer (10 mM Tris pH 7.5; 140 mM NaCl; 1 mM EDTA; 0.2 mg/ml), during 30 minutes

of incubation. A 100 µl solution of 2 pM DNA prepared with above method, having digoxigenin at one end and biotin at other, prepared in TNEB was injected followed by 20 minutes of incubation. A further wash with TNEB removed untethered DNA. Thereafter, 2 µl of M280 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 free floating biotinylated end of tethered DNA. While 3 µm M280 streptavidin beads were used to probe palindromic hairpin dynamics, a 1 µm Streptavidin C1 beads (ThermoFisher, 65001) were used in flow-stretching essay.

A further elimination of non-specifically attached magnetic beads was performed by applying 20 pN of force for M280 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.

Force on the magnetic beads were measured using variance method, following the correction free framework of Ostrofet et al.[24] 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 4kb DNA to moderate-force regime to avoid high-force denaturation and pursue force range of interest between 1 to 20 pN.

Substrate construction

The 1kb DNA containing a 60nt palindromic hairpin was assembled from three parts. Two separate PCRs were performed on a template plasmid to create the two 500bp 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 4kbp DNA was created from three parts. A plasmid was digested with BsaI (NEB) to create the central 4kbp section with two non-complementary overhangs. Separately two PCRs were performed with the same plasmid as a template to create two 500bp 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 4kbp section. This created a DNA with a total length of 5kbp with 500bp of one end being randomly interspersed with biotin and the other end with digoxigenin.

Author contributions

J.L. and A.S. contributed equally. J.L. wrote the Python and MATLAB code. A.S., J.L, 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 experimental 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

This work was supported by National Institutes of Health grant CA067007 and AI150496 (R.F.). Support for T.S. was provided by 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.

References

1. Neuman, K.C. & Nagy, A. Single-molecule force spectroscopy: optical tweezers, magnetic tweezers and atomic force microscopy. *Nature Methods* **5**, 491-505 (2008).
2. Gosse, C. & Croquette, V. Magnetic tweezers: micromanipulation and force measurement at the molecular level. *Biophys J* **82**, 3314-29 (2002).
3. De Vlaminck, I. & Dekker, C. Recent advances in magnetic tweezers. *Annu Rev Biophys* **41**, 453-72 (2012).
4. Dulin, D. An Introduction to Magnetic Tweezers. in *Single Molecule Analysis : Methods and Protocols* (eds. Heller, I., Dulin, D. & Peterman, E.J.G.) 375-401 (Springer US, New York, NY, 2024).
5. Dulin, D. et al. High Spatiotemporal-Resolution Magnetic Tweezers: Calibration and Applications for DNA Dynamics. *Biophys J* **109**, 2113-25 (2015).
6. 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).
7. Agarwal, R. & Duderstadt, K.E. Multiplex flow magnetic tweezers reveal rare enzymatic events with single molecule precision. *Nature Communications* **11**, 4714 (2020).
8. Rueden, C.T. et al. ImageJ2: ImageJ for the next generation of scientific image data. *BMC Bioinformatics* **18**, 529 (2017).
9. 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).
10. 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).
11. Huhle, A. et al. Camera-based three-dimensional real-time particle tracking at kHz rates and Ångström accuracy. *Nature Communications* **6**, 5885 (2015).
12. Saha, P. et al. Exploring force-driven stochastic folding dynamics in mechano-responsive proteins and implications in phenotypic variation. *Nature Communications* **16**, 1018 (2025).
13. 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).
14. 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).
15. Morgan, I.L. & Saleh, O.A. Tweezepy: A Python package for calibrating forces in single-molecule video-tracking experiments. *PLoS One* **16**, e0262028 (2021).
16. Susano Pinto, D.M. et al. Python-Microscope - a new open-source Python library for the control of microscopes. *J Cell Sci* **134**(2021).
17. Rueden, C.T. et al. PyImageJ: A library for integrating ImageJ and Python. *Nature Methods* **19**, 1326-1327 (2022).

18. Phillips, M.A. et al. Microscope-Cockpit: Python-based bespoke microscopy for bio-medical science. *Wellcome Open Res* **6**, 76 (2021).
19. Harris, C.R. et al. Array programming with NumPy. *Nature* **585**, 357-362 (2020).
20. Nishino, R. & Loomis, S.H.C. Cupy: A numpy-compatible library for nvidia gpu calculations. *31st confernce on neural information processing systems* **151**(2017).
21. Virtanen, P. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods* **17**, 261-272 (2020).
22. 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).
23. Pineda, J. et al. Geometric deep learning reveals the spatiotemporal features of microscopic motion. *Nature Machine Intelligence* **5**, 71-82 (2023).
24. Ostrofet, E., Papini, F.S. & Dulin, D. Correction-free force calibration for magnetic tweezers experiments. *Scientific reports* **8**, 15920 (2018).

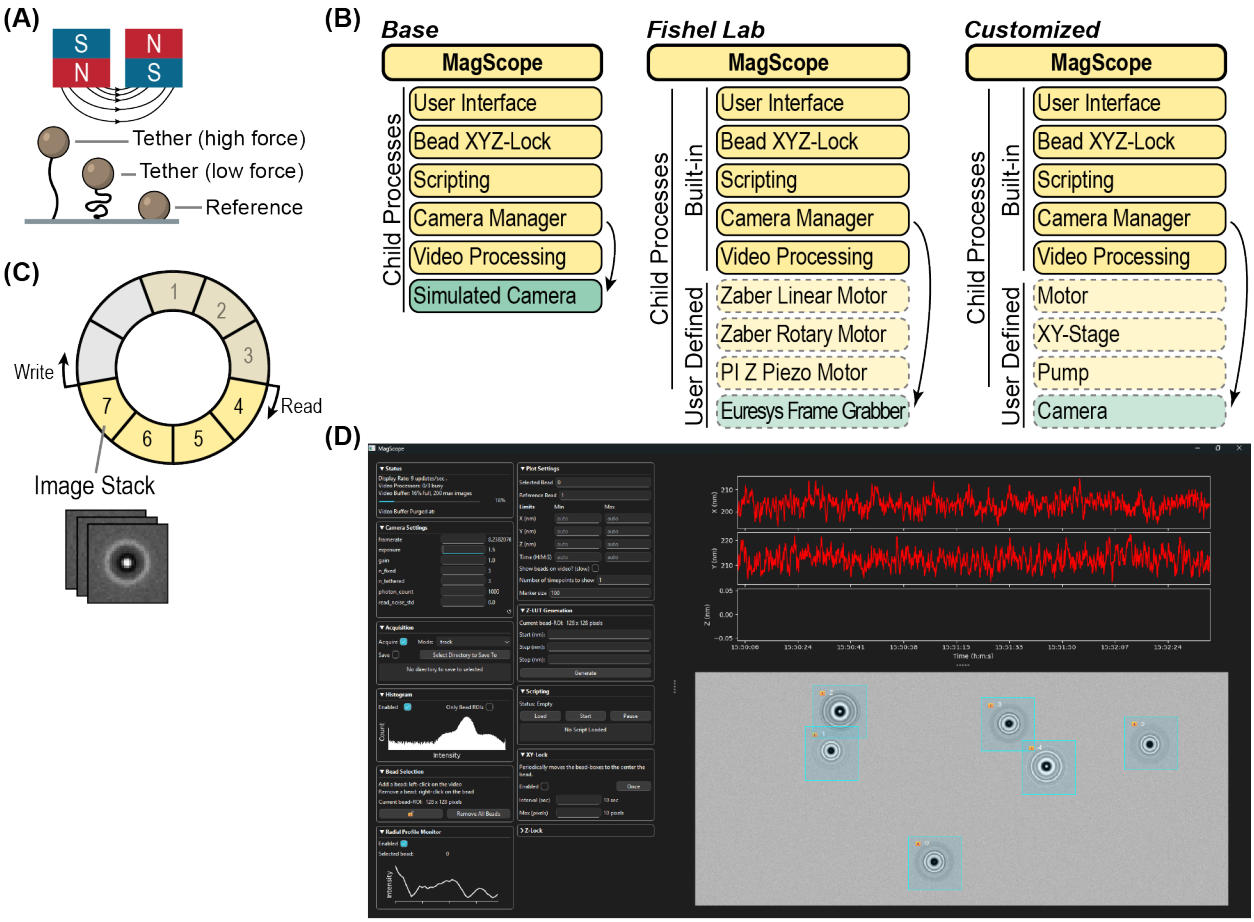

Figure 1. Overview of MagScope

(A) Cartoon overview of magnetic tweezers principle. **(B)** 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. **(C)** Diagram of how ring-buffers work to provide efficient inter-process access to large datasets including the video stream. **(D)** Screenshot of MagScope base user interface.

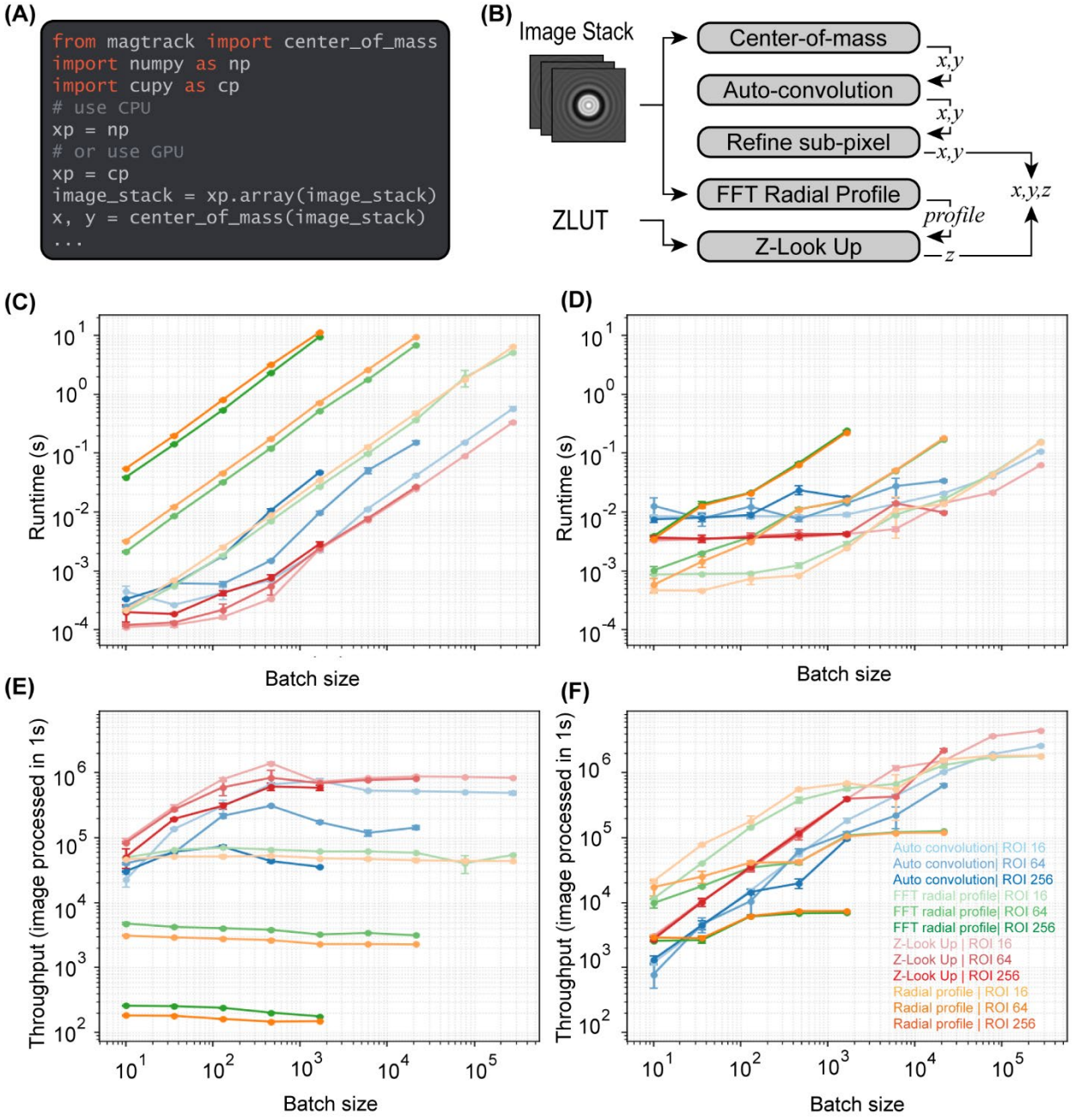

Figure 2. Overview of MagTrack

(A) Example of coding with MagTrack for a standard center-of-mass calculation. **(B)** Overview of tracking pipelines that can be composed with MagTrack. **(C)** CPU runtime of various MagTrack algorithms on images stacks with a width and height of 16, 64 or 256, for various image stack batch sizes. **(D)** Same as (C) but on a GPU. **(E)** Same as (C) but the CPU throughput in 1 second. **(F)** Same as (E) but on a GPU.

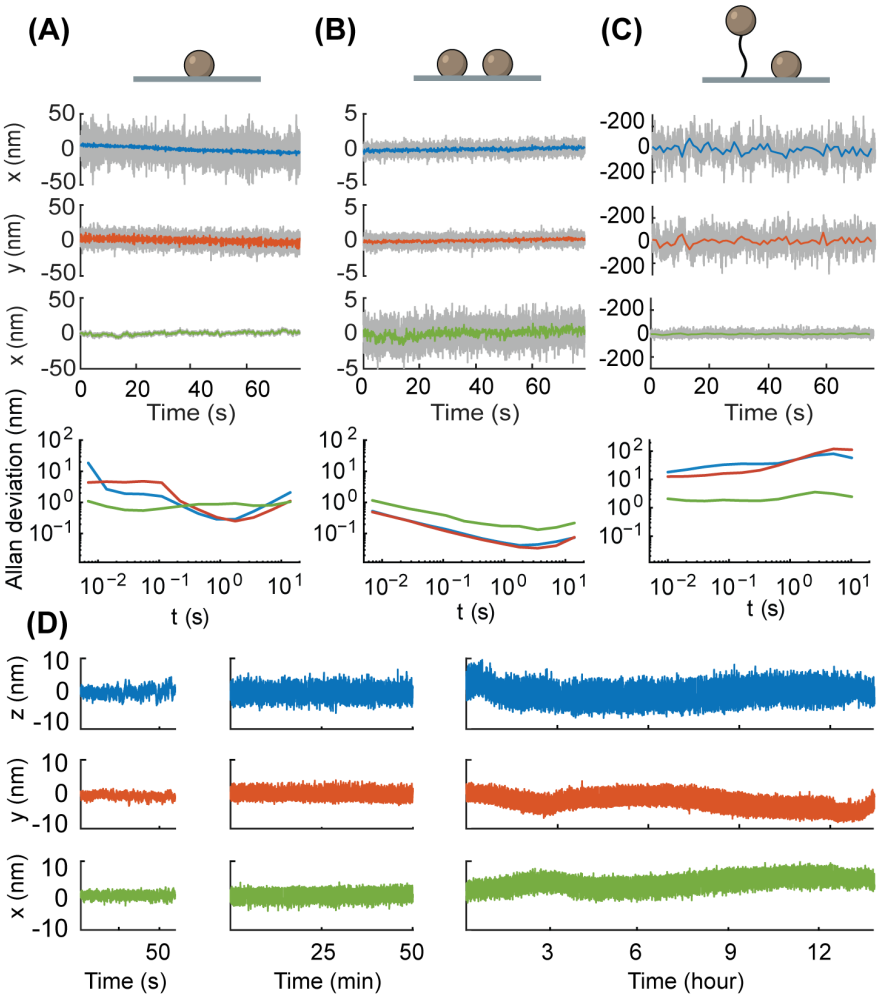

Figure 3. Positional stability in x, y, and z and Allan deviation

(A) Raw 100 Hz (gray) and 1 Hz downsampled (color) x, y, and z traces of a fixed bead; and Allan deviation corresponding to each trace. **(B)** Traces for fixed bead subtracted from another fixed bead. **(C)** Traces for a single DNA tethered magnetic bead subtracted from a fixed bead held at 1 pN of force. **(D)** Traces of a fixed bead subtracted by another fixed bead in x, y, and z for 1 min, 1 hour, and 14 hours at 100 Hz.

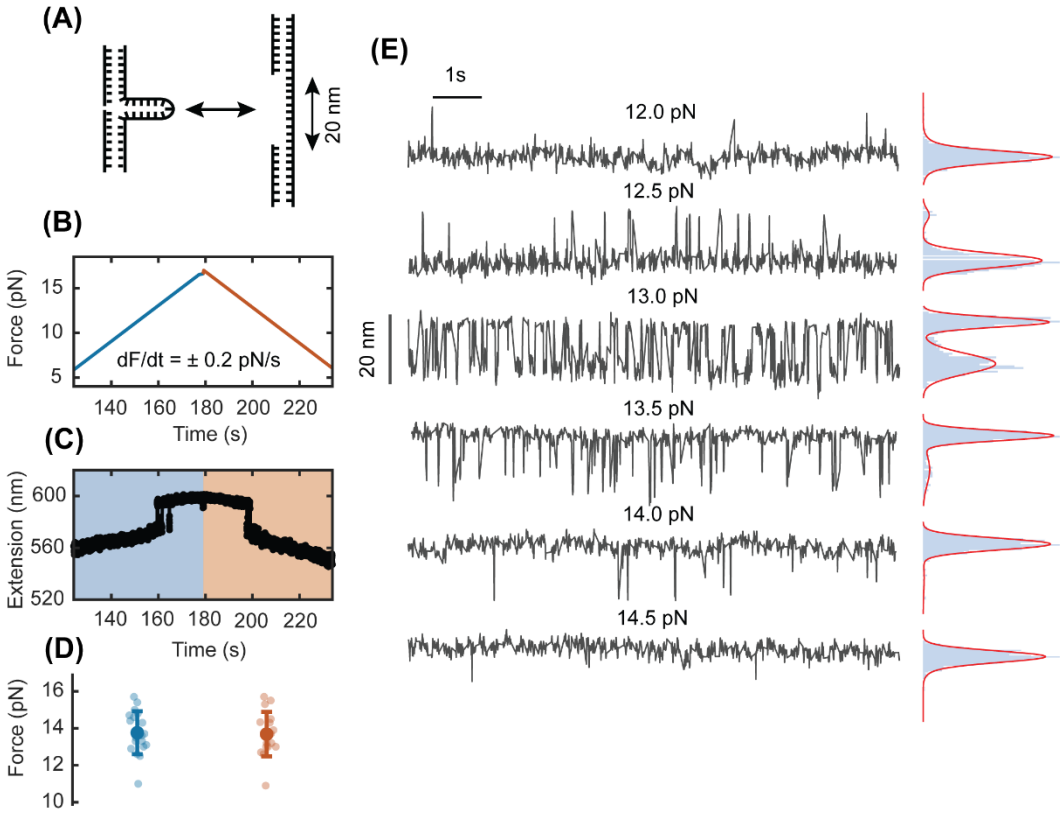

Figure 4. Dynamics of DNA hairpin folding

(A) Schematic of a reversible two-state DNA hairpin. **(B)** Force ramp from 6pN to 16pN at 0.2 pN/s. **(C)** Representative trace of a DNA extension of hairpin shown in **A** over a force ramp shown in **B**. Hairpin ruptures at ~160 s and reforms at ~200 s. **(D)** Distribution of rupture and reformation force of 18 molecules with mean (solid circle) and standard deviation (error bar). **(E)** Traces of DNA extensions of hairpin in **A** at the indicated forces (12.0-14.5 pN), occupancy histograms with kernel density estimates. The population shifts from predominantly closed to predominantly open with increasing force, with comparable occupancy near ~13 pN which is identified as rupture force.

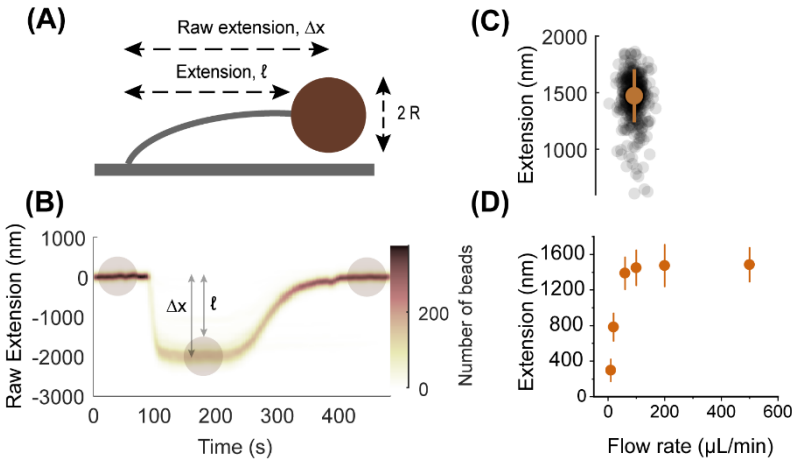

Figure 5. Flow stretching 1,000 molecules

(A) Assay schematic: 4kb DNA tethered to glass surface at one end and to a 1 μm magnetic bead at the other, a moderate magnetic force of ~1pN persists in lateral dimension throughout the assay while flow applied the hydrodynamic drag. **(B)** 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. The kernel-density was mapped for the accepted 375 molecules, selected from 1,000 initially tracked (10x objective; 28x28-pixel ROIs; 100 Hz). **(C)** Absolute DNA extension distribution for the QC-accepted subset (N = 375) after correcting panel **B** traces by subtracting the bead radius (~500 nm); marker shows mean and standard deviation. **(D)** Mean DNA extension at different flow rates is shown. Extension approaches the 4 kb contour length (~1.3-1.4 μm) by ~200 $\mu L \cdot min^{-1}$ and remains stable at higher rates.