Review of Scientific Instruments

RESEARCH ARTICLE | MAY 06 2024

A neural network-based algorithm for the reconstruction and filtering of single particle trajectory in magnetic particle tracking

Mohit Prashanth ; Pan Du ; Jian-xun Wang ; Huixuan Wu

Rev. Sci. Instrum. 95, 055001 (2024)
https://doi.org/10.1063/5.0183533

Review of Scientific Instruments ARTICLE pubs.aip.org/aip/rsi

A neural network-based algorithm for the reconstruction and filtering of single particle trajectory in magnetic particle tracking

Cite as: Rev. Sci. Instrum. 95, 055001 (2024); doi: 10.1063/5.0183533
Submitted: 23 October 2023 • Accepted: 23 April 2024 •
Published Online: 6 May 2024

Mohit Prashanth,[1] Pan Du,[2] Jian-xun Wang,[2] and Huixuan Wu[3,a)]

AFFILIATIONS

[1] Aerospace Engineering, University of Kansas, Lawrence, Kansas 66045, USA
[2] Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, Indiana 46556, USA
[3] Mechanical Engineering Department, Florida State University, Tallahassee, Florida 32310, USA

[a)] **Author to whom correspondence should be addressed:** hwu8@fsu.edu

ABSTRACT

Magnetic particle tracking (MPT) is a recently developed non-invasive measurement technique that has gained popularity for studying dense particulate or granular flows. This method involves tracking the trajectory of a magnetically labeled particle, the field of which is modeled as a dipole. The nature of this method allows it to be used in opaque environments, which can be highly beneficial for the measurement of dense particle dynamics. However, since the magnetic field of the particle used is weak, the signal-to-noise ratio is usually low. The noise from the measuring devices contaminates the reconstruction of the magnetic tracer's trajectory. A filter is then needed to reduce the noise in the final trajectory results. In this work, we present a neural network-based framework for MPT trajectory reconstruction and filtering, which yields accurate results and operates at very high speed. The reconstruction derived from this framework is compared to the state-of-the-art extended Kalman filter-based reconstruction.

I. INTRODUCTION

Particle trajectories provide critical information for the study of particle dynamics in multiphase or granular flows. Optical-based particle tracking techniques are state-of-the-art methods.[1,2] However, they require the systems under investigation to be transparent. Dense particulate flow systems, such as batch mixing systems[3] in chemical, food, and pharmaceutical sectors, and industrial systems, such as fluidized beds[4] and rotating drums,[5] usually operate in opaque environments. The inability of optical-based methods to be applied to opaque environments has been the driving factor for developing particle tracking techniques for opaque systems, including radioactive particle tracking (RPT),[6] positron emission particle tracking (PEPT),[7,8] and magnetic resonance imaging (MRI).[9] These methods have both advantages and drawbacks. For instance, PEPT is a popular and reliable tracking technique with high accuracy and time resolution,[10] which is vital for studying rapid dynamics.[11] It is often used as a validation tool for theoretical and numerical models such as the discrete element method (DEM) and computational fluid dynamics (CFD) in granular systems.[12,13] Recently, a machine learning algorithm was developed to identify and track multiple particles in PEPT that achieved a spatial resolution of 2 mm.[14] Despite PEPT's reliability, it does require expensive equipment, and the handling of radioactive materials demands special expertise. On the other hand, the MRI approach does not rely on radioactive materials, but its temporal resolution is usually insufficient for high-speed dynamics.[15]

Magnetic particle tracking (MPT) offers an economic solution for particle tracking in opaque environments,[16,17] as it does not need any special equipment for radiation safety. It uses a magnetically labeled particle, some magnetometers, and a computer for data processing. With MPT, the presence of magnetic poles makes it possible to obtain the tracer's rotational information,[18] which is difficult to achieve using other techniques. This is vital for understanding the dynamics governing granular motion.[19,20] The MPT method was introduced in the field of medicine to examine the gastrointestinal tract without using x rays.[21] Since then, numerous methods have been developed to solve the MPT problem and reconstruct a

tracer's trajectory for a wide range of applications. Analytical solutions are developed to locate a magnetic tracer in real time,[21] which is useful in areas such as targeted drug delivery and classification of buried mines.[22] An early analytical solution to MPT was a simple formula that was based on the tracer's magnetic field and its gradients.[23] Despite this method's simplicity, a special sensing device was required that can measure both the magnetic field and its gradients. Improved analytical solutions include the scalar triangulation and ranging (STAR) method[22] and its derivatives,[24–26] which address the issue of the asphericity error encountered in the STAR method. A limitation of these analytical methods is that they require gradients of the field and depend on a specific sensor arrangement. Other quick and robust analytical or semi-algebraic solutions that do not rely on the gradients of magnetic field are proposed in Refs. 27 and 28. Although fast in processing data, these algebraic and analytical methods usually possess no filtering or optimization functions that can mitigate the fluctuation caused by measurement uncertainty.

Optimization algorithms have been developed for MPT to achieve accurate results for studying complex physics in fluidized beds,[29,30] stirred media,[31] and rotating equipment.[32,33] The formulation of loss functions in these algorithms[18,34] enables an arbitrary arrangement of sensors, which is not generally seen in analytical methods. With this arrangement, an array of sensors can be used to obtain field data with good signal-to-noise ratio (SNR), allowing for highly accurate reconstruction of particle trajectories.[29,35] However, finding a global minimum with these algorithms can be time consuming and, in some cases, may not even be possible if the initialization is poor. Noisy reconstruction from the optimization algorithm can be smoothed using the Kalman filter.[36] Köhler[37] applied this methodology to study fuel mixing in a fluidized bed, revealing the presence of three fuel segregation regimes for the operating ranges investigated. In fact, the particle trajectory reconstruction can be modeled as a state-space problem and solved the using extended Kalman filter (EKF), which has been shown to achieve the same accuracy as optimization methods but with greater processing speed.[38]

The performance of a reconstruction method also depends on its ability to reduce noise. Traditional reconstruction and filtering methods, such as EKF, usually assume that noise has certain mathematical features (e.g., noise is at high frequency or follows Gaussian distribution), which cannot address certain systematic errors in a specific sensor setup. Thanks to the development of neural networks (NN), a new kind of filter has emerged. The NN-based methods, such as the convolutional neural network (CNN) and recurrent neural network (RNN), have the capability to learn specific patterns in a sequence of signal and are increasingly recognized in the field of signal processing, e.g., filtering electroencephalogram (EEG) signals,[39] electrocardiogram (ECG) signals,[40,41] radar signals,[42] and speech recognition and separation.[43,44] An early work in employing NN as a filter integrates wavelet filters into it to remove high-frequency noise in ECG signals.[45] Since then, NNs have shown capability in filtering noise containing multiple characteristic frequencies, including drifting noise that can be difficult to filter using methods such as wavelet.[46] NNs can also learn the sparse information present in the time–frequency decomposition of a noisy signal and filter out band-limited, low-frequency, and cyclic noise.[47] In addition, NN-based filters have demonstrated the ability to preserve phase information in the input signal while reducing mutual interference noise, making them highly valuable in multiple radar sensor systems for autonomous driving.[48] Specialized RNN networks, such as the Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM), have the advantage to recognize and remember long-term correlations in a time sequence, which is critical to denoise the sequence.[28,49–53]

In this work, we present NN-based particle tracking and denoising algorithms to reconstruct the trajectory of a magnetic tracer in MPT. Our approach utilizes a multi-layer perceptron (MLP) network for the initial reconstruction of the tracer's position. MLP networks are known as "universal approximator" due to their ability to learn any mapping function.[54,55] In our work, the MLP network learns the non-linear mapping between magnetic field signals and the tracer's trajectory. In addition, we employ a GRU-based RNN to reduce the noise in the MLP-reconstructed trajectories. Similarly, we constructed an MLP-GRU framework for orientation reconstruction. Using separate networks was inspired by the study of Wu *et al.*[28] They demonstrated the capability of using separate networks for denoising position and orientation of a single magnetic tracer for simulated datasets.

This paper is organized as follows: Sec. II outlines the principles underlying the MPT method. Subsequently, Sec. III provides insights into the experimental setup. Section IV focuses on three reconstruction methods employed in our study: image-based, EKF-based, and NN-based, along with parametric studies conducted to select crucial parameters for NN training. Finally, Sec. V delves into the performance of the NN framework and its comparison with the state-of-the-art EKF method.

II. MAGNETIC DIPOLE AND MPT PRINCIPLE

To perform MPT, a small magnetic source moving in a flow field and a few magnetic sensors (magnetometers) are required, as shown in Fig. 1. The magnetic field can be modeled using the dipole equation,

$$\boldsymbol{B}(\boldsymbol{r}, \boldsymbol{r}_0, \boldsymbol{m}) = \frac{\mu_0}{4\pi} \frac{[3\boldsymbol{n} \cdot (\boldsymbol{m} \cdot \boldsymbol{n}) - \boldsymbol{m}]}{|\boldsymbol{r} - \boldsymbol{r}_0|^3}, \tag{1}$$

where, $\boldsymbol{r_0}$ is the location of a magnetometer, $\boldsymbol{r}$ is the magnet's position, $\boldsymbol{n}$ is the normal vector in the direction $\boldsymbol{r} - \boldsymbol{r_0}$, $\boldsymbol{m}$ is the magnetic moment, and μ_0 is permeability of air. The magnetometers placed in a known configuration record the field strength D_i,

$$D_i = \boldsymbol{B}_i(r, r_{0i}, m) \cdot \boldsymbol{S}_i, \tag{2}$$

where $\boldsymbol{B}$ is determined using [Eq. (1)], $\boldsymbol{S_i}$ is the orientation of the ith (i = 1, 2, . . ., N) sensor in the lab frame of reference, and N is the total number of sensors. It is necessary to know beforehand the location and orientation of the sensors relative to the reference coordinate system. The task of MPT is to determine the position and orientation of the magnetic source, constituting its trajectory, at every instant, according to the field strength measured by the magnetometers. For this purpose, we developed a framework based on neural network algorithms and compared them to the EKF method.

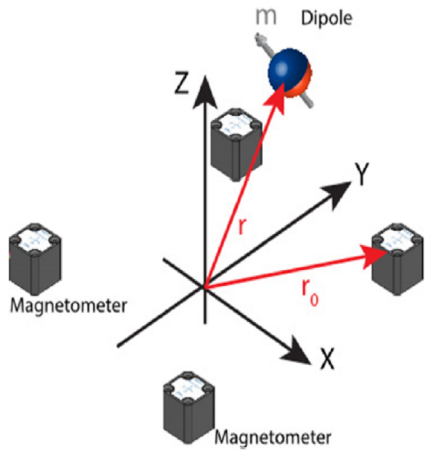


FIG. 1. Schematics of MPT.

III. EXPERIMENTAL SETUP

The experimental setup, as shown in Fig. 2(a), consisted of a circular tube through which air flowed upward into the domain of interest (a cubical glass enclosure with a mesh on top), 4 three-channel magnetometers (constituting 12 measurement channels), two high-speed cameras, and a spherical ball with a magnet located at its center. In the following text, we refer to this ball as the tracer. 65 cfm (cubic feet/min) of air from a compressor flowed through a vertical tube of diameter 9.5 cm and height 90 cm. The upward flow suspended the tracer in the air and caused it to move randomly in the glass enclosure, which was further amplified by its collisions with the walls. The enclosure had dimensions of $8 \times 8 \times 8$ cm^3 with a mesh fixed on top to keep the tracer inside the enclosure.

We placed two cameras 90 cm from the center of the experimental setup to record the 3D movement of the tracer in the domain. They were high-speed cameras, NAC HX-5, with Tokina 100 mm F2.8 lenses. The cameras were set to capture motion at a frame rate of 500 Hz with a resolution of 1088×1032 pixel^2. The stereo angle between the cameras was 90°, allowing for the easy projection of pixels to the lab frame of reference. This setup enabled the image-based 3D reconstruction of trajectories, which was independent of the MPT reconstruction. As discussed later, images captured by the cameras were processed to obtain the magnet's position and orientation at every instant, and these results served as the ground truth for MPT calibration or validation.

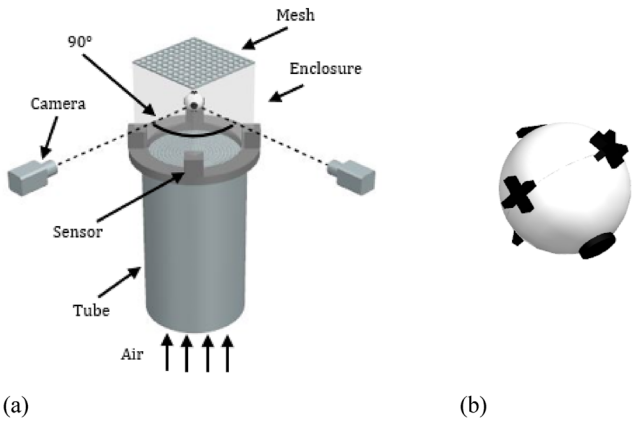


FIG. 2. (a) Experiment setup and (b) 3D printed tracer with the magnet at its center.

TABLE I. Location of sensors in the lab frame of reference.

	x (m)	y (m)	z (m)
Sensor No. 1	0.045	0.045	0.01
Sensor No. 2	−0.045	−0.045	0.01
Sensor No. 3	−0.045	0.045	0.01
Sensor No. 4	0.045	−0.045	0.01

Four 3-axis fluxgate Bartington M612 magnetometers were placed in a circular configuration surrounding the enclosure. The locations of the magnetometers were known in the lab frame of reference and are presented in Table I. Each magnetometer has 3 channels and recorded the field strength in 3 directions, totaling 12 sensor readings at any instant. The magnetometers recorded the magnetic field strength of the moving magnetic tracer, which was then used as input to the EKF and NN algorithms for trajectory reconstruction. These magnetometers have a sensing range of (−90, 90 μT) and a maximum sampling rate of 3000 Hz. Finally, the cameras and magnetometers were synchronized and set to a sampling rate of 500 Hz.

In our work, a magnet was positioned at the center of a 3D-printed sphere (printed using polylactic acid) with a diameter of 2 cm, serving as the magnetic tracer. This cylindrical magnet had a moment of 0.0125 Am^2 and was made of neodymium rare earth with dimensions of 1/16 × 1/32 in. To determine the magnet's orientation at any given moment, six markers with different shapes ("O," "–," "Y," "+," "★," and "*") were printed on the tracer's surface. The dimensions of the markers, in terms of circumscribed circles, were 6 mm in diameter with 1 mm outward protrusion. The 3D design of the tracer with the markers attached is shown in Fig. 2(b). Markers "O" and "–" indicated the north and south poles of the magnet, respectively. The magnetic moment aligned with the direction of marker "O." The remaining four markers ("Y," "+," "★," and "*") were evenly distributed around the equator of the tracer. This arrangement allowed us to employ vector cross product to determine the orientation of the tracer at any instant.

IV. RECONSTRUCTION METHODS

Formulation of the reconstruction methods: image, EKF, and NN-based, are described in this section. Reconstruction obtained the images that served as the ground truth in our work, and the EKF reconstruction served as an indicator to gauge the performance of the NN framework through direct result comparison.

A. Image reconstruction

The images captured by the cameras were processed to obtain the position and orientation of the magnetic tracer in three dimensions. Two sets of images at every instant, one from each camera, allowed for 3D reconstruction. The results obtained from the

images were used to supervise the training of the neural networks. The image processing performed in MATLAB R2022a involved the following steps:

1. Background subtraction: the background was recorded before we introduced the ball into the domain. During image processing, we subtracted this background from the raw image. For example, Fig. 3(a) shows a raw image obtained by one of the cameras. Once the background is removed, only the tracer is retained in the image, as shown in Fig. 3(b).
2. Binarization: after background subtraction, the image matrix was then converted to a binary matrix of 0s and 1s representing dark and bright regions in the image, respectively. This was done using an in-built MATLAB adaptive thresholding function that chooses threshold values based on the local mean intensity in each pixel's neighborhood. Intensity values below the threshold were changed to 0 and above threshold to 1.
3. Center identification: at any instant, determining the true position of the magnet meant locating the center of the tracer in the image. For this, we used an in-built function called "imfindcircles" in MATLAB to detect the center. This function is based on the circular Hough transform, which detects the outline of a circle and subsequently derives its center and radius.
4. Orientation vector determination: as the first step of orientation reconstruction, the image matrix of the tracer in the binary image [in Fig. 3(b)] was inverted (changing 0s to 1s and 1s to 0s). This resulted in the markers becoming bright and other regions inside the tracer becoming dark [as shown in Fig. 3(c)], allowing us to determine the centroid of the markers (using the center of mass method in the Matlab function, "regionprops"). Note that the circular outline of the tracer is added in Fig. 3(c) for representation only. This outline was not present in image processing. Consequently, the magnet's orientation vector was obtained by one of the following three ways:
    - The simplest case was that the North Pole (represented by marker "O") was present in the camera's field of view [Fig. 3(c)]. For such cases, the orientation vector was obtained by determining the vector from the center of the tracer to the center of the marker. This case is shown in Fig. 3(c), where the center of the ball is shown as the red asterisk, and the center of the marker "O" is shown as the blue asterisk, along with the vector between them.
    - If the South Pole (represented by the marker "–") was in the field of view, the orientation vector was obtained by simply taking the vector from the center of the marker to the center of the ball.
    - If neither "O" nor "-" was in the camera's field of view, we made use of the markers that were present on the equator of the tracer ("Y," "+," "★," and "*"). They were perpendicular to the moment of the magnet. One marker was selected from each image (totaling to two markers from two images at any instant in time). Both the markers selected were the closest to the center of the tracer in the image. Subsequently, we could obtain two vectors perpendicular to the north direction of the magnet. The cross product of these vectors gave us the orientation vector.

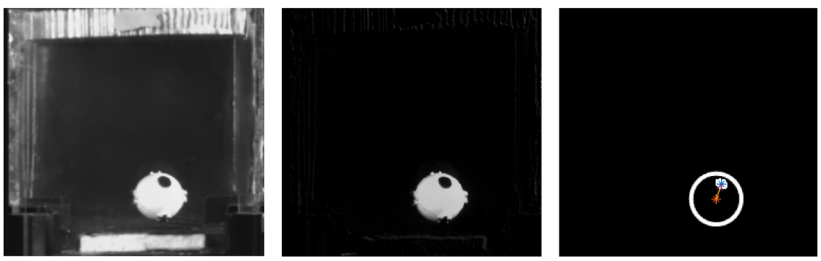
B. EKF reconstruction

Tao *et al.*[38] demonstrated that the MPT problem can be framed as a state-space model and solved using EKF. We adopted the same formulation in this study. We represent the state (unknown) variables as $\boldsymbol{X}_k = \begin{pmatrix} \boldsymbol{r}_k \\ \boldsymbol{m}_k \end{pmatrix}$, where $\boldsymbol{r}$ denotes the tracer's position and $\boldsymbol{m}$ represents its moment at the current time step k. Equation (3) presents the kinematic equation, which models the tracer's current position. This equation relies on the tracer's position and velocity at the previous time step $k-1$, denoted by $\boldsymbol{r}_{k-1}$ and $\boldsymbol{u}_{k-1}$, respectively,

$$\boldsymbol{r}_k = \boldsymbol{r}_{k-1} + \boldsymbol{u}_{k-1}\Delta t + \boldsymbol{w}_k^r. \tag{3}$$

The moment of the magnetic tracer $\boldsymbol{m}_k$ was calculated using a quaternion as shown in Eq. (4). The quaternion $\boldsymbol{q}$, which signifies the tracer's rotation, is expressed by Eq. (5), where Ω represents the magnitude of the tracer's angular velocity and $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ denotes the unit vector along the axis of rotation. Both tracer velocity, $\boldsymbol{u}$, and quaternion, $\boldsymbol{q}$, can be estimated using historical data,

$$\boldsymbol{m}_k = \boldsymbol{q}_{k-1}\boldsymbol{m}_{k-1}\boldsymbol{q}_{k-1}^{-1} + \boldsymbol{w}_k^m, \tag{4}$$



(a) (b) (c)

FIG. 3. Steps in the processing of images. (a) Sample image from the experimental dataset, (b) background subtraction, and (c) processed image showing the vector from the tracer center to the marker center.

$$\boldsymbol{q} = \left(\cos\frac{\Omega\Delta t}{2}, \omega_x \sin\frac{\Omega\Delta t}{2}, \omega_y \sin\frac{\Omega\Delta t}{2}, \omega_z \sin\frac{\Omega\Delta t}{2}\right). \quad (5)$$

In Eqs. (3) and (4), $\boldsymbol{w}_k^r$ and $\boldsymbol{w}_k^m$ are independent Gaussian variables that depict the unknown perturbations in the tracer's position and orientation. These uncertainty variables are assumed to possess a zero mean, with their covariance represented by Q_k.

Henceforth, we derive the forward model F of the state variables $\boldsymbol{X}_k$ by combining Eqs. (3) and (4),

$$\boldsymbol{X}_k = \begin{pmatrix} \boldsymbol{r}_k \\ \boldsymbol{m}_k \end{pmatrix} = F(\boldsymbol{X}_{k-1}) + \boldsymbol{w}_k, \quad (6)$$

where

$$F(\boldsymbol{X}_{k-1}) = \begin{pmatrix} \boldsymbol{r}_{k-1} + \boldsymbol{u}_{k-1}\Delta t \\ \boldsymbol{q}_{k-1}\boldsymbol{m}_{k-1}\boldsymbol{q}_{k-1}^{-1} \end{pmatrix} \quad (7)$$

and

$$\boldsymbol{w}_k = \begin{pmatrix} \boldsymbol{w}_k^r \\ \boldsymbol{w}_k^m \end{pmatrix}.$$

The measurements acquired by the ith sensor at time step k are described by Eq. (8). Here, D represents the theoretical, noise-free measurement of the magnetic field strength, as defined in Eq. (2). In addition, $\boldsymbol{v}_k$ represents another independent variable that accounts for uncertainties in the measurements, assumed to have a zero mean. The covariance matrix for $\boldsymbol{v}_k$ is denoted by $\boldsymbol{R}_k$,

$$\begin{aligned} O_{i,k} &= D_i(\boldsymbol{X}_k)(1 + \boldsymbol{v}_k), \\ D_i &= \boldsymbol{B}(\boldsymbol{X}_k) \cdot \boldsymbol{S}_i, \end{aligned} \quad (8)$$

Equations (6) and (8) collectively constitute the state-space model. Within the EKF procedure, determining the conditional probability of the state variable $\boldsymbol{X}_k$, given the measurements O_k, is necessary. We denote this conditional probability as $P(\boldsymbol{X}_k|O_k)$. In this model, $P(\boldsymbol{X}_k|O_k)$ is approximated by a Gaussian distribution that can be completely characterized by its mean and covariance.

Determining the state variables at time step k involves two steps, prediction and correction, as follows:

a. In the prediction step, a prediction of the mean and covariance of the state variables given by $\boldsymbol{X}_k^-$ and P_k^- is made using Eq. (9). In this equation, $\boldsymbol{X}_{k-1}$ and P_{k-1} represent the mean and covariance obtained at time step $k-1$, respectively, and F_k denotes the Jacobian of the model function F evaluated at $\boldsymbol{X}_{k-1}$,

$$\begin{aligned} \boldsymbol{X}_k^- &= F(\boldsymbol{X}_{k-1}), \\ P_k^- &= F_k P_{k-1} F_k^T + Q_k, \end{aligned} \quad (9)$$

where

$$F_k = \frac{\partial F}{\partial X}(\boldsymbol{X}_{k-1}).$$

b. In the correction step, the predicted state and covariance $\boldsymbol{X}_k^-$ and P_k^- are updated to $\boldsymbol{X}_k$ and P_k using the measurement O_k. This update is given by the Kalman formula as follows:

$$\begin{aligned} \boldsymbol{X}_k &= \boldsymbol{X}_k^- + K_K(O_k - D(\boldsymbol{X}_k^-)), \\ P_k &= (I - K_k H_k)P_k^-, \end{aligned} \quad (10)$$

where

$$\begin{aligned} K_k &= P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R_k V_k^T\right)^{-1}, \\ H_k &= \frac{\partial D}{\partial X}(\boldsymbol{X}_k^-), \\ V_k &= \frac{\partial O_k}{\partial v_k}. \end{aligned}$$

Here, H_k and V_k are Jacobian matrices, and K_k is the standard EKF correction matrix (for example, see Refs. 38 and 56 for more details).

C. Neural networks construction and training

We used PyTorch 2.0.0 to develop the NN model, and the networks were trained on the Nvidia 2080 Graphics Processing Unit (GPU). Two sets of networks were developed, one for position and another for orientation reconstruction. Each set contains an MLP reconstruction network and an RNN denoiser.

1. Reconstruction network

The reconstruction of a trajectory aims to find a mapping from the measurements to the particle position and orientation. MLP network, a universal approximator, provides a straightforward model. Therefore, our initial reconstruction is performed using MLP networks.

a. Dataset and preprocessing. The time series dataset in our work has the form $\boldsymbol{D} = (\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ represents the sensor readings and $\boldsymbol{y}$ denotes the image-based trajectory reconstruction [Eq. (11)], which is used as ground truth for network training,

$$\begin{aligned} \boldsymbol{x} &= \begin{bmatrix} x_1^1 \cdots x_1^n \cdots x_1^{3N} \\ \vdots \quad \ddots \quad \vdots \\ x_T^1 \cdots x_T^n \cdots x_T^{3N} \end{bmatrix}, \\ \boldsymbol{y} &= \begin{bmatrix} r_{x,1}, r_{y,1}, r_{z,1} \\ \vdots \\ r_{x,T}, r_{y,T}, r_{z,T} \end{bmatrix} \textit{for position network}, \\ \boldsymbol{y} &= \begin{bmatrix} \theta_{x,1}, \theta_{y,1}, \theta_{z,1} \\ \vdots \\ \theta_{x,T}, \theta_{y,T}, \theta_{z,T} \end{bmatrix} \textit{for orientation network}, \end{aligned} \quad (11)$$

where x_t^n means the signal from the nth magnetic sensor at time t. T means the total time steps, and $3N$ is the total number of sensor readings. The entries in $\boldsymbol{y}$ are the position (r_x, r_y, r_z) or orientation $(\theta_x, \theta_y, \theta_z)$ at a certain time. The position vectors $\boldsymbol{r}$ are measured relative to the center of the experimental domain. The orientation vectors $\boldsymbol{\theta}$ are unit vectors. The complete dataset consisted of $T = 280\,000$ datapoints. Half of them, i.e., 140 000 datapoints, were used for developing networks. Once trained, the networks were applied on the remaining 140 000 datapoints in the dataset to obtain the trajectory.

Before the sensor data were fed into the reconstruction network for training, we preprocessed and normalized them in the following steps:

1. Remove background Earth field: during the data collection process, sensor readings were recorded prior to the introduction of the tracer into the experimental domain. These recorded readings are labeled as $\boldsymbol{x}_{earth}$, which are subsequently removed from the real measurements [Eq. (12)],

$$\boldsymbol{x} \leftarrow (\boldsymbol{x} - \bar{\boldsymbol{x}}_{earth}) * \eta, \tag{12}$$

$\bar{\boldsymbol{x}}_{earth}$ is the mean of $\boldsymbol{x}_{earth}$. Factor $\eta = 1000/89$ converts the recorded sensor readings into Volts to field strength in micro-Tesla.

2. Normalization: to ensure that the sensor readings have the same order of magnitude, they were scaled to have zero mean and unit standard deviation using Eq. (13a). Here, $\bar{\boldsymbol{x}}$ and $\boldsymbol{\mu}_x$ are the mean and standard deviation calculated over each sensor reading in the dataset, respectively. Similarly, $\boldsymbol{y}$ was scaled to zero mean and unit standard deviation [Eq. (13b)] using scaling factors $\bar{\boldsymbol{y}}$ and $\boldsymbol{\mu}_y$ computed in the same manner,

$$\boldsymbol{X}^n \leftarrow \frac{\boldsymbol{x}^n - \overline{\boldsymbol{x}^n}}{\mu_x^n}, . \tag{13a}$$

$$\boldsymbol{Y}^k \leftarrow \frac{\boldsymbol{y}^k - \overline{\boldsymbol{y}^k}}{\mu_y^k}. \tag{13b}$$

Given the preprocessed dataset, the problem of reconstruction can be posed as finding the non-linear mapping function between the normalized sensor readings $\boldsymbol{X}$ and ground truth $\boldsymbol{Y}$.

b. Network architecture. An MLP network consists of an input layer, an output layer, and hidden layers between them. An element in the hidden layer (known as the neuron) takes the inputs x from the input layer (or previous layer of neurons), sums them up with weights w and a bias b [Eq. (14a)], and finally generates an output y using a non-linear rectified linear unit (ReLU) function [Eq. (14b)],

$$z = \sum_{i=1}^{P} w_i x_i + b, \tag{14a}$$

$$y = ReLU(z). \tag{14b}$$

Here, P in Eq. (14a) represents the number of neurons in the preceding layer. The ReLU function is $ReLU(z) = z$ if $z \geq 0$ and $ReLU(z) = 0$ if $z < 0$. It introduces nonlinearity in the network. The outputs y can be the final output of the network or an input to the following layers. The parameters w and b are determined in the training process explained in the following. A network with proper parameters can approximate any continuous function.[54,55]

c. Network training process. The network is trained by adaptively adjusting the weights w and biases b to minimize a loss function that measures the difference between the ground truth and the network predictions. The loss function in our network is the averaged L^2 norm [Eq. (15)], i.e., the mean squared error (MSE) between the network predictions $\boldsymbol{Y}_{pred}$ and ground truth (image-based results) $\boldsymbol{Y}$,

$$MSE_{w,b}(\boldsymbol{Y}, \boldsymbol{Y}_{pred}) = \frac{\sum_{t=1}^{T} (\boldsymbol{Y} - \boldsymbol{Y}_{pred})^2}{T}. \tag{15}$$

As aforementioned, the position and orientation reconstructions use different networks. In the position network, $\boldsymbol{Y}$ is the normalized (r_x, r_y, r_z), and in the orientation network, $\boldsymbol{Y}$ is $(\theta_x, \theta_y, \theta_z)$. The training processes are the same for both networks. It includes forward pass, backward pass, and parameter (weights and biases) update. In the forward pass, the network takes the inputs, calculates the predictions using Eqs. (14a) and (14b), and evaluates the error using Eq. (15). Backward pass involves determining the gradient of the loss with respect to all the weights and biases in the network. In the MLP network, this process is called backpropagation (BP). The gradients are used to adjust the network parameters. The adjustments are done using an optimization procedure called adaptive moment estimation (ADAM). The adjusted parameters produce a lower MSE. These adjustments are controlled using a learning rate (LR) to ensure stable training. A high LR speeds up the training process at the cost of accuracy. A low LR may lead to better accuracy but at the expense of slow network training. In our work, the network is trained until it starts to overfit the training dataset. Overfitting can be determined by evaluating the network on a separate dataset, called validation dataset, during each iteration of training. Generally, a network is overfit when it fits the training dataset with high accuracy but performs poorly on the validation dataset.

2. Denoiser network

Trajectories reconstructed with the MLP networks are further denoised with GRU-based RNNs. We term the GRU networks as denoiser networks. Their architecture and training process are described in this section.

a. Dataset. We concatenated the MLP-reconstructed trajectories $\boldsymbol{x}'$ with the image-reconstructed trajectories to form the dataset $\boldsymbol{D}' = (\boldsymbol{x}', \boldsymbol{y}')$. The image-reconstructed $\boldsymbol{y}'$ were used to determine the error in the network predictions and supervise the training. Note that $\boldsymbol{y}'$ here is the same as the normalized signal $\boldsymbol{Y}$ in Eq. (13b), but we use a new symbol to indicate that they are used for the denoiser network. For position denoising, $\boldsymbol{y}'$ is the normalized (r_x, r_y, r_z), and for orientation, $\boldsymbol{y}'$ is $(\theta_x, \theta_y, \theta_z)$. The dataset for the denoiser networks contained 140 000 datapoints, of which 125 000 were used for training, and the remaining 15 000 for fine-tuning and validating the denoiser networks.

b. Gated recurrent unit (GRU) network architecture. Different from MLP, the hidden layers of an RNN can store information from the past in the form of a hidden state. This allows an RNN to capture temporal dependencies in a time series. In a general RNN, the inputs at current time x_t along with the hidden state from the previous time step h_{t-1} are used to update the hidden state h_t [Eq. (16)]. Subsequently a prediction y_t is made using Eq. (17),

$$h_t = f(W_h h_{t-1}, W_x x_t, b_h, b_x), \tag{16}$$

$$y_t = W_o h_t + b_o. \tag{17}$$

Here, h, W, and b represent the hidden state, network weights, and biases, respectively. Subscripts t, h, x, and o pertain to the time step, hidden state, input, and output, respectively. The function f is non-linear. Different fs are used in different RNN models.

GRU network is an enhanced version of RNN. A GRU network uses gating structures to control the flow of information. The specific algorithm is listed in Eqs. (18)–(21). The update gate z_t determines how much information in the past should be used at time step t [Eq. (18)]. The matrices W_{xz} and W_{hz} contain the weight parameters. On the other hand, the reset gate r_t determines how much information in the previous time step matters at the current time step, t [Eq. (19)]. If the information stored in the hidden state h_{t-1} is uncorrelated to the input at the current time step, the reset gate becomes zero and erases the contribution from h_{t-1}. This is shown in Eq. (20) in the form of Hadamard product of the reset gate r_t and hidden state h_{t-1}, in the determination of a candidate hidden state $\tilde{h}_t$. A linear interpolation of the candidate state $\tilde{h}_t$ and previous state h_{t-1} determines the current hidden state [Eq. (21)]. This interpolation is the key component for long term dependencies in the time series data. If the update gate z_t is close to one, the hidden state update is dependent more on the current input and the recent past, whereas if the update gate is close to zero, then the hidden state is not updated,

$$z_t = \sigma\left(W_{xz}x_t + W_{hz}h_{t-1} + b_z\right), \tag{18}$$

$$r_t = \sigma\left(W_{xr}x_t + W_{hr}h_{t-1} + b_r\right), \tag{19}$$

$$\tilde{h}_t = \tanh\left(W_{x\tilde{h}}x_t + W_{h\tilde{h}}(r_t \cdot h_{t-1}) + b_{\tilde{h}}\right), \tag{20}$$

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t. \tag{21}$$

Subscripts r, z, and $\tilde{h}$ pertain to reset gate, update gate, and candidate hidden state, respectively; σ is the non-linear sigmoid function $\sigma(x) = 1/(1 + e^{-x})$; and tanh is the hyperbolic tangent function. The final output of the network is given by Eq. (17).

c. Network training process. The training process of GRU requires a more complex loss function and a backpropagation depending on history. The loss function employed in the denoiser networks considers the smoothness of the trajectory. It includes two terms [Eq. (22)]: the error of direct network predictions, MSE_{pred} [Eq. (15)], and that of smoothness, MSE_{grad} [Eq. (23)]. The smoothness is defined as the finite difference (gradient) between successive predictions [Eq. (24)]. In addition, $\dot{\boldsymbol{y}}$ and $\dot{\boldsymbol{y}}_{pred}$ denote the gradients of the ground truth and network's predictions, respectively. The smoothness of the tracer's position can be simply understood as velocity (displacement per time step). The parameter λ in Eq. (22) is the weighting factor controlling the significance of MSE_{grad}. It needs to be optimized, as specified later. This formulation of loss function aids the minimization of error in the network's predictions and the error in the gradients of the predictions simultaneously during the training process. The loss at each time step is evaluated and summed up for all time steps,

$$Loss = MSE_{pred} + \lambda * MSE_{grad}, \tag{22}$$

$$MSE_{grad} = MSE(\dot{\boldsymbol{y}}, \dot{\boldsymbol{y}}_{pred}), \tag{23}$$

$$\dot{\boldsymbol{y}}_t = \boldsymbol{y}_{t+1} - \boldsymbol{y}_t. \tag{24}$$

During the backpropagation, GRU must consider the hidden states. The gradients of loss function with respect to the weights and biases pertaining to the hidden states and inputs at each time step are evaluated, in what is called the backpropagation through time (BPTT). The process of BPTT can become demanding in terms of computation and memory when training the network with a single and long sequence of data.

Truncated BPTT is often employed to reduce the computational and memory cost by reducing the number of time steps for which the loss is backpropagated.[57,58] This may not always be practical as crucial information might be available in the time steps that are truncated. Instead in our work, to manage the computational and memory demands, we divided the training dataset into shorter sequences and fed batches of the sequences to the denoiser networks as input, similar to the method in Ref. 59.

D. Network parametric study

In this section, we outline the methodology to optimize the trajectory reconstruction and denoising networks. The hyperparameters employed in the MLP and denoiser networks were determined through heuristic methods and are presented in Table II. Our focus here is to determine the ideal dataset size for effective training of the reconstruction networks, pinpoint the optimal input sequence length for the denoiser networks to minimize prediction errors, and explore the impact of λ in [Eq. (22)] to account for errors in gradients of denoiser network's predictions. It is important to note that we evaluated the network performance on validation sets using the MSE metric [Eq. (15)].

1. Training dataset size vs reconstruction network performance

First, we focused on the optimal number of datapoints for effective MLP network training. We conducted a series of trainings, gradually increasing the size of the training dataset from 25 000 to 150 000 datapoints. To assess the performance of the networks, we employed a validation set, comprising 15 000 datapoints. It is worth noting that all the training datasets underwent normalization using the mean and standard deviation derived from the largest dataset, which contained 150 000 points. This allowed us to make meaningful comparison of errors from networks trained with different dataset sizes.

TABLE II. Hyperparameters of reconstruction and denoiser networks.

Network settings	Reconstruction	Denoiser
Weight decay	1×10^{-5}	1×10^{-6}
Learning rate (LR)	0.001	0.001
Layers	5	3
Neurons	50	10
Batches	10	5-6

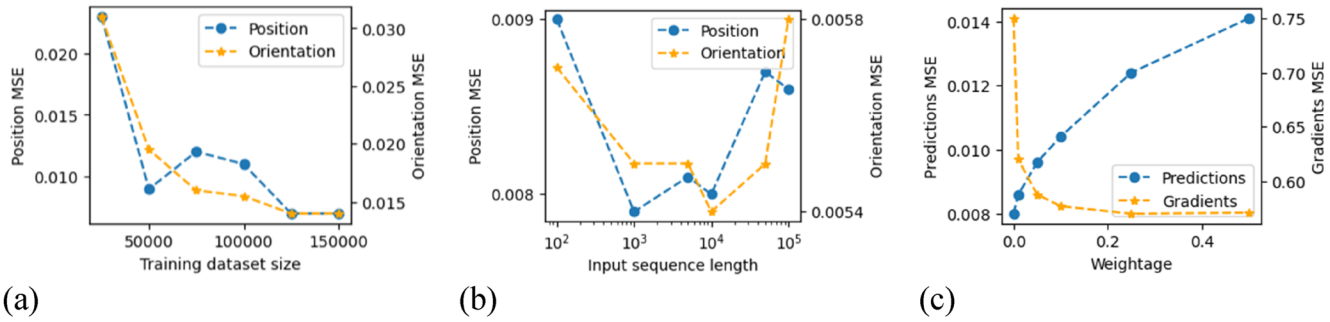


FIG. 4. (a) Effect of the training dataset size on reconstruction networks, (b) effect of the input sequence length on denoiser networks, and (c) effect of the gradient weighting factor on the position denoiser network.

Figure 4(a) shows the errors of position and orientation networks with different training datasets. As the size of the training data increases, the error in the predictions decreases as expected, indicating an improving performance. As a side note, the position result [blue line shown in Fig. 4(a)] has fluctuations (at 50 000 training size), which is common in NN training [60] and is probably due to the randomness of data shuffling. No considerable improvement is seen from 125 000 to 150 000 datapoints. Consequently, we opted to utilize the networks trained with 125 000 datapoints for the trajectory reconstruction.

2. Input sequence length vs denoiser network performance

The training dataset for GRU consisted of 125 000 datapoints. Using the entire training data as a large single input sequence for the GRU is computationally very demanding and may not necessarily improve the network performance. Hence, we conducted a study to assess the performance of the GRU network as a function of increasing input sequence lengths: 100, 1000, 5000, 10 000, 50 000, and 100 000 datapoints. Their performance was assessed on a validation set containing 15 000 datapoints. Figure 4(b) shows the error in the predictions of position and orientation denoising. The error in network predictions exhibited a "U" shape with the minimum located at 10 000. The network performance declines for input sequences larger than 10 000 points. This can be attributed to the fact that as the input sequence length increases, the network must form and hold memory in its computational unit for longer sequences. This may not always be advantageous in terms of the network's performance.[59,61,62]

3. Gradient weightage vs denoiser network performance

The loss function in denoiser networks includes a parameter λ, controlling the weight of trajectory gradients or smoothness [Eq. (22)]. Here, we use the results from position denoiser network to demonstrate the effect of λ. Six different values, ranging from 0 to 0.5, were selected for this investigation. The validation sets used in this study were the same as those used in the previous study regarding input sequence length. The MSE in predictions and MSE in gradients are depicted using blue and orange curves shown in Fig. 4(c), respectively. As the weightage of gradients in the loss function increases, the error in the gradients of network's predictions decreases, but concurrently, the error in the predictions increases. In other words, a large λ value results in a smoother trajectory with lower position accuracy. The MSE_{grad} reaches a plateau for $\lambda > 0.1$, while the prediction error becomes worse. The crossing point of MSE_{pred} and MSE_{grad} is a good option for both low prediction error and high smoothness. Therefore, we choose $\lambda = 0.05$ in the modified loss function.

To summarize the training process in this framework, we utilized a dataset comprising a total of 280 000 datapoints. Of these, 125 000 datapoints were dedicated to training the reconstruction networks. An additional 15 000 datapoints were set aside for fine-tuning the network and validation purposes. Subsequently, the trained networks were leveraged to reconstruct the remaining 140 000 datapoints. The reconstructed trajectories along with their corresponding image-reconstructed counterparts were employed to train the denoiser networks. Of those, 15 000 points were used to fine-tune and validate the denoiser networks. The remaining 125 000 points were segmented into sequences of 10 000 datapoints, serving as input for the denoiser networks. The loss function was designed to take into account the error in the gradients of network's prediction. The weightage of the gradient error was set to be 0.05. Finally, we evaluated this framework on an independent testing set consisting of 5000 points, the results of which are detailed in Sec V.

TABLE III. Computation time for training the networks.

	Position		Orientation	
Networks	MLP	GRU	MLP	GRU
Time (s)/epoch	0.022	14	0.022	14
No. of epochs	210 000	1100	7 000	4 755
Total time (s)	4 620	15 400	154	66 570
Total training time (s)	20 020		66 724	
Total training time (h)	~5.5		~18.5	

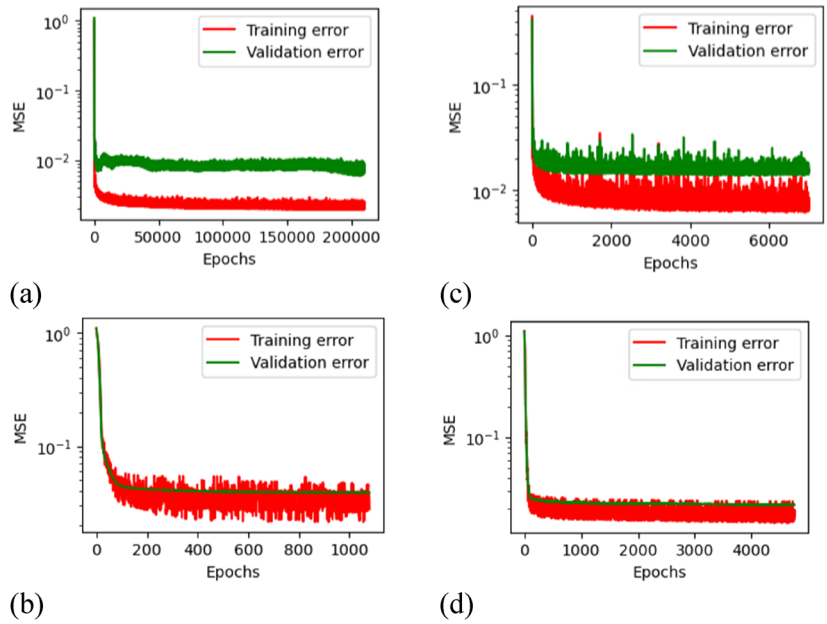


FIG. 5. (a) Training plot of position reconstruction network, (b) training plot of the position denoiser network, (c) training plot of the orientation reconstruction network, and (d) training plot of the orientation denoiser network.

V. RESULTS AND DISCUSSION

In this section, we present the results obtained from testing our NN framework on an independent dataset. The training process of the position and orientation networks is shown in Fig. 5, showcasing the error in network predictions as a function of epochs (or iterations) during training for both training and validation datasets. It is worth noting that all the networks were trained until they exhibited initial signs of overfitting the training data.

Table III presents the time required for training these networks. Notably, the reconstruction networks completed an epoch (or iteration) in ~0.022 s, while the denoiser networks required significantly more time, i.e., around 14 s/epoch. In total, the position networks finished training in ~5.5 h, with the orientation networks demanding much more time at 18.5 h. This slower training pace (i.e., high number of epochs) can be attributed in part to a low LR of 0.001. Despite the extended training times, the trained networks exhibited remarkable speeds in processing the data. Testing on an independent dataset comprising 5000 points, we employed the "process_time_ns" function from the "time" package in Python to precisely measure the time consumption. The combined testing time for both the position and orientation networks on the independent dataset is presented in Table IV. The processing speed of the reconstruction networks was less than 0.02 ms/point. On the other hand, both position and orientation denoiser networks required 1.75 s each to process the testing dataset, totaling to just 3.5 s/5000 points or 0.7 ms/point.

Figures 6 and 7 show sample trajectories from the testing dataset reconstructed using image-, EKF-, and NN-based methods. Figure 6 shows the tracer's position (r_x, r_y, r_z) over time, with position vectors relative to the experimental domain's center.

TABLE IV. Computation time for testing the networks.

Networks	MLP	GRU
Time (ms/point)	≤0.02	0.7

Meanwhile, Fig. 7 shows the tracer's orientation in the x, y, and z directions $(\theta_x, \theta_y, \theta_z)$ with time, represented as the components of a unit vector. Upon observing these figures, it becomes evident that the NN reconstruction closely aligns with the results obtained from EKF and image reconstruction methods, indicating a strong comparative performance.

To evaluate the uncertainties δ, in EKF and NN-based reconstructions, we calculated the mean absolute error (MAE) between their predictions and image-based reconstruction,

$$\delta_r = \frac{\sum_{t=1}^{T} |\boldsymbol{r}_t - \boldsymbol{r}_{image,t}|}{T}, \qquad \delta_\theta = \frac{\sum_{t=1}^{T} |\boldsymbol{\theta}_t - \boldsymbol{\theta}_{image,t}|}{T}, \tag{25}$$

where $\boldsymbol{r}$ and $\boldsymbol{\theta}$ are the 3D position and orientation vectors of the magnetic tracer, respectively, t indicates the current time step, and T is the total time steps. In addition, δ_r and δ_θ represent the uncertainty in reconstruction of the tracer's position and orientation,

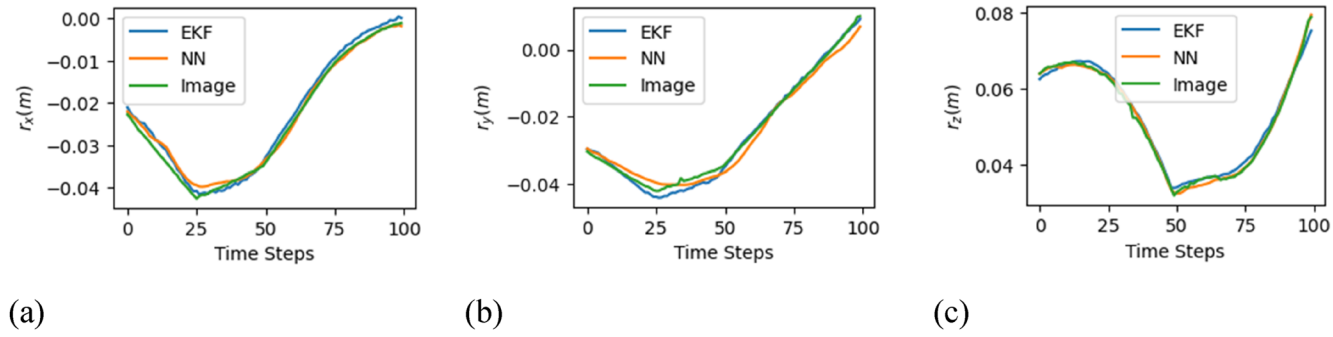


FIG. 6. (a) Reconstructed *x* position, (b) reconstructed *y* position, and (c) reconstructed z position.

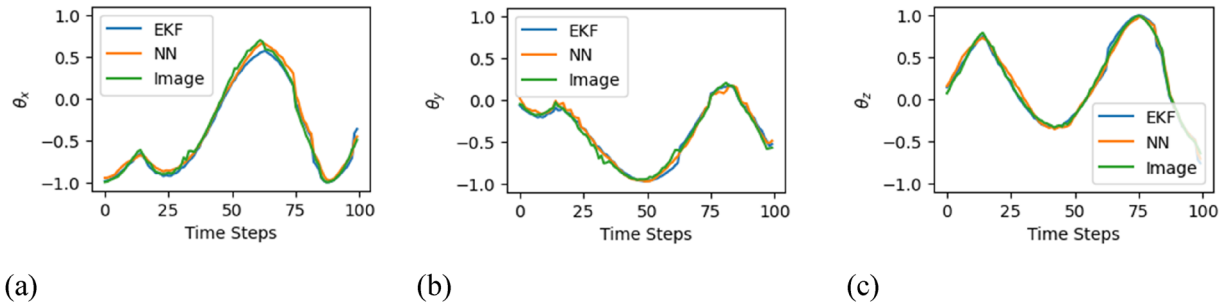


FIG. 7. (a) Orientation of the tracer in *x* direction, (b) orientation of the tracer in *y* direction, and (c) orientation of the tracer in *z* direction.

respectively. In addition, we also determined the uncertainty in the gradients of the trajectories given by

$$\delta_{\dot{\boldsymbol{r}}} = \frac{\sum_{t=1}^{T-1} \left|\dot{\boldsymbol{r}}_t - \dot{\boldsymbol{r}}_{image,t}\right|}{T-1}, \qquad \delta_{\dot{\boldsymbol{\theta}}} = \frac{\sum_{t=1}^{T-1} \left|\dot{\boldsymbol{\theta}}_t - \dot{\boldsymbol{\theta}}_{image,t}\right|}{T-1}, \tag{26}$$

where $\dot{\boldsymbol{r}}$ and $\dot{\boldsymbol{\theta}}$ are the gradients of position and orientation of the magnetic tracer, respectively. They are calculated using forward finite difference [Eq. (24)]. A comparison between the uncertainty in EKF and NN-based reconstruction gives us an insight into how the NN framework performs relative to the state-of-the-art EKF method. Table V presents this comparison. In addition, uncertainty in the position was normalized by the dimensions of the experimental domain (80 mm in each direction), whereas uncertainty in the orientation of the tracer was converted to degrees. These are denoted as normalized position error and orientation error (deg), presented in Table VI. Note that the EKF noise in this study is slightly larger compared to that of our previous work in Ref. 38. This is due to the higher electromagnetic interference caused by the electric air pump.

From Tables V and VI, we find the following:

(a) Normalized uncertainty in position was 1.25% of the domain size in the EKF approach, while it was 1.4% for our NN-based approach. The EKF method has a slightly better position reconstruction. We attribute the higher uncertainty of the NN method to the fact that this method considers the smoothness of the trajectory through the application of λ in [Eq. (22)], which compromises the position accuracy.

(b) Uncertainty in EKF-reconstructed orientation was 2.8°, whereas it was 2.4° in NN-reconstructed orientation. The latter method is better.

(c) The position gradient $\delta_{\dot{r}}$ results suggest that the NN model has a lower uncertainty.

TABLE V. Comparison of uncertainties in EKF and NN-based reconstruction.

Method	δ_r (m)	$\delta_{\dot{r}}$ (m)	δ_θ (rad)	$\delta_{\dot{\theta}}$ (rad)
EKF	0.001	3.6×10^{-4}	0.049	0.022
NN	0.0011	3.4×10^{-4}	0.042	0.025

TABLE VI. Comparison of normalized position error (%) and orientation error (deg).

Method	δ_r (%)	δ_θ
EKF	1.25	2.8°
NN	1.4	2.4°

The MPT method aims to measure the velocity and orientation of a particle in an opaque environment, which are important to study granular dynamics. The above-mentioned results show that

the NN model did slightly better in reconstructing particle velocity and orientation.

VI. CONCLUSION

In this study, we introduced an innovative approach that employed neural networks to reconstruct a tracer's trajectory based on the magnetic field strength data. We developed two distinct sets of neural networks: one dedicated to reconstructing the tracer's position and the other focused on reconstructing its orientation. Each set consisted of a combination of an MLP network for the initial reconstruction and a GRU-based RNN for denoising the MLP results. The training process for both the position and orientation networks was completed in a day. Once trained, the networks processed the testing dataset at speeds significantly higher than the sensor sampling rate (which was 500 Hz) in the validation experiment. This feature allows for the real-time reconstruction of MPT trajectories at a sampling rate of up to 1400 Hz (on a computer with Nvidia 2080). This aspect is particularly valuable in applications demanding live monitoring of tracer particles, where immediate and accurate tracking is essential.

We showed that the NN method with systematically chosen parameters can reconstruct particle trajectories. The NN results are comparable with the state-of-the-art EKF reconstruction. Moreover, our neural network framework guarantees excellent performance when the reconstruction problem is well-posed, i.e., the number of sensors is larger than the degree of freedom of the particle. Compared to traditional methods, NN-based models have another advantage that they can recognize the pattern of biased noise and reduce it accordingly.

The current NN framework can only reconstruct a single particle trajectory. To extend its capabilities to multi-particle tracking systems, adjustments are necessary: the output neuron vector in the networks should be modified as a concatenation of multiple particle states. Meanwhile, the uniqueness of the output vector should be assured as the permutation of particles will result in various output vectors that represent the same states of multiple particles. We can solve this by specifying the order of the particles, such as ranking them according to momentum strength. It is also worth noting that our data-driven NN-based approach prefers interpolation rather than extrapolation, that is, a significant alteration in the sensor positions necessitates re-training and testing of the networks. For example, to track a particle in a much larger system, we need to use more sensors and the network needs re-training. In addition, the signal-to-noise ratio is usually low in a large setup. A magnetic shield should be used to improve the data quality.

ACKNOWLEDGMENTS

The project was supported by the NSF (Grant No. 2403832).

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Mohit Prashanth: Data curation (equal); Methodology (equal); Writing – original draft (equal). **Pan Du**: Investigation (equal); Validation (equal); Writing – review & editing (equal). **Jian-xun Wang**: Funding acquisition (equal); Supervision (equal). **Huixuan Wu**: Conceptualization (equal); Funding acquisition (equal); Supervision (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]R. Ni, S. Kramel, N. T. Ouellette, and G. A. Voth, "Measurements of the coupling between the tumbling of rods and the velocity gradient tensor in turbulence," J. Fluid Mech. **766**, 202–225 (2015).
[2]G. A. Voth and A. Soldati, "Anisotropic particles in turbulence," Annu. Rev. Fluid Mech. **49**(1), 249–276 (2017).
[3]A. Guida, A. W. Nienow, and M. Barigou, "Lagrangian tools for the analysis of mixing in single-phase and multiphase flow systems," AIChE J. **58**(1), 31–45 (2011).
[4]C. T. Crowe, J. D. Schwarzkof, M. Sommerfeld, and Y. Tsuji, *Multiphase Flows with Droplets and Particles* (CRC Press, 2012).
[5]W. Rong, Y. Feng, P. Schwarz, P. Witt, B. Li, T. Song, and J. Zhou, "Numerical study of the solid flow behavior in a rotating drum based on a multiphase CFD model accounting for solid frictional viscosity and wall friction," Powder Technol. **361**, 87–98 (2020).
[6]J. S. Lin, M. M. Chen, and B. T. Chao, "A novel radioactive particle tracking facility for measurement of solids motion in gas fluidized beds," AIChE J. **31**(3), 465–473 (1985).
[7]D. J. Parker, C. J. Broadbent, P. Fowles, M. R. Hawkesworth, and P. McNeil, "Positron emission particle tracking—A technique for studying flow within engineering equipment," Nucl. Instrum. Methods Phys. Res., Sect. A **326**(3), 592–607 (1993).
[8]B. P. B. Hoomans, J. A. M. Kuipers, M. A. Mohd Salleh, M. Stein, and J. P. K. Seville, "Experimental validation of granular dynamics simulations of gas-fluidised beds with homogenous in-flow conditions using positron emission particle tracking," Powder Technol. **116**(2–3), 166–177 (2001).
[9]M. Nakagawa, S. A. Altobelli, A. Caprihan, E. Fukushima, and E. K. Jeong, "Non-invasive measurements of granular flows by magnetic resonance imaging," Exp. Fluids **16**, 54–60 (1993).
[10]D. J. Parker, R. N. Forster, P. Fowles, and P. S. Takhar, "Positron emission particle tracking using the new Birmingham positron camera," Nucl. Instrum. Methods Phys. Res., Sect. A **477**(1–3), 540–545 (2002).
[11]T. Pöschel and N. Brilliantov, *Granular Gas Dynamics* (Springer, Berlin/Heidelberg, 2003).
[12]A. Hassanpour, H. Tan, A. Bayly, P. Gopalkrishnan, B. Ng, and M. Ghadiri, "Analysis of particle motion in a paddle mixer using discrete element method (DEM)," Powder Technol. **206**(1–2), 189–194 (2011).
[13]J. H. Ferziger, M. Perić, and R. L. Street, *Computational Methods for Fluid Dynamics* (Springer, 2019).
[14]A. L. Nicuşan and C. R. Windows-Yule, "Positron emission particle tracking using machine learning," Rev. Sci. Instrum. **91**(1), 013329 (2020).
[15]R. Stannarius, "Magnetic resonance imaging of granular materials," Rev. Sci. Instrum. **88**(5), 051806 (2017).
[16]A. Köhler, D. Pallarès, and F. Johnsson, "Magnetic tracking of a fuel particle in a fluid-dynamically down-scaled fluidised bed," Fuel Process. Technol. **162**, 147–156 (2017).
[17]P. Jayaprakash, "*Granular hydrodynamic study of non-spherical particles in a 3D fluidized bed*," Master's dissertation, Eindhoven University of Technology, 2016.
[18]I. Mema, K. A. Buist, J. A. M. Kuipers (Hans), and J. T. Padding, "Fluidization of spherical versus elongated particles: Experimental investigation using magnetic particle tracking," AIChE J. **66**(4), e16895 (2019).

[19]I. Mema, V. Mahajan, B. Fitzgerald, H. Kuipers, and J. Padding, "Effect of lift force on dense gas-fluidized beds of non-spherical particles," in *Proceedings of the 12th International Conference on Computational Fluid Dynamics in the Oil and Gas, Metallurgical and Process Industries* (SINTEF, 2017), Vol. 2, pp. 71–79.
[20]I. Mema, V. V. Mahajan, B. W. Fitzgerald, and J. T. Padding, "Effect of lift force and hydrodynamic torque on fluidisation of non-spherical particles," Chem. Eng. Sci. **195**, 642–656 (2019).
[21]W. Andrä, H. Danan, W. Kirmße, H.-H. Kramer, P. Saupe, R. Schmieg, and M. E. Bellemann, "A novel method for real-time magnetic marker monitoring in the gastrointestinal tract," Phys. Med. Biol. **45**(10), 3081–3093 (2000).
[22]R. F. Wiegert, "Magnetic STAR technology for real-time localization and classification of unexploded ordnance and buried mines," Proc. SPIE **7303**, 73031U (2009).
[23]T. Nara, S. Suzuki, and S. Ando, "A closed-form formula for magnetic dipole localization by measurement of its magnetic field and spatial gradients," IEEE Trans. Magn. **42**(10), 3291–3293 (2006).
[24]Y. Sui, G. Li, S. Wang, and J. Lin, "Asphericity errors correction of magnetic gradient tensor invariants method for magnetic dipole localization," IEEE Trans. Magn. **48**(12), 4701–4706 (2012).
[25]H. H. Jin, Z. H. Zhuang, and H. B. Wang, "None-asphericity-error method for magnetic dipole target detection," IEEE Geosci. Remote Sens. Lett. **15**(8), 1294–1298 (2018).
[26]G. Yin, P. Li, Z. Wei, G. Liu, Z. Yang, and L. Zhao, "Magnetic dipole localization and magnetic moment estimation method based on normalized source strength," J. Magn. Magn. Mater. **502**, 166450 (2020).
[27]K. A. Buist and T. M. J. Nijssen, "Magnetic particle tracking: A semi-algebraic solution," Chem. Eng. Sci. **265**, 118212 (2023).
[28]H. Wu, P. Du, R. Kokate, and J.-X. Wang, "A semi-analytical solution and AI-based reconstruction algorithms for magnetic particle tracking," PLoS One **16**(7), e0254051 (2021).
[29]K. A. Buist, A. C. van der Gaag, N. G. Deen, and J. A. Kuipers, "Improved magnetic particle tracking technique in dense gas fluidized beds," AIChE J. **60**(9), 3133–3142 (2014).
[30]L. Yang, J. T. Padding, K. A. Buist, and J. A. M. Kuipers, "Three-dimensional fluidized beds with rough spheres: Validation of a two fluid model by magnetic particle tracking and discrete particle simulations," Chem. Eng. Sci. **174**, 238–258 (2017).
[31]A.-C. Böttcher, C. Schilde, and A. Kwade, "Experimental assessment of grinding bead velocity distributions and stressing conditions in stirred media mills," Adv. Powder Technol. **32**(2), 413–423 (2021).
[32]T. M. Nijssen, M. A. H. van Dijk, H. A. Kuipers, J. van der Stel, A. T. Adema, and K. A. Buist, "Experiments on floating bed rotating drums using magnetic particle tracking," AIChE J. **68**(5), e17627 (2022).
[33]L. Zhang, F. Weigler, V. Idakiev, Z. Jiang, L. Mörl, J. Mellmann, and E. Tsotsas, "Experimental study of the particle motion in flighted rotating drums by means of magnetic particle tracking," Powder Technol. **339**, 817–826 (2018).
[34]J. Neuwirth, S. Antonyuk, S. Heinrich, and M. Jacob, "CFD–DEM study and direct measurement of the granular flow in a rotor granulator," Chem. Eng. Sci. **86**, 151–163 (2013).
[35]K. A. Buist, P. Jayaprakash, J. A. M. Kuipers, N. G. Deen, and J. T. Padding, "Magnetic particle tracking for nonspherical particles in a cylindrical fluidized bed," AIChE J. **63**(12), 5335–5342 (2017).
[36]E. Sette, D. Pallarès, F. Johnsson, F. Ahrentorp, A. Ericsson, and C. Johansson, "Magnetic tracer-particle tracking in a fluid dynamically down-scaled bubbling fluidized bed," Fuel Process. Technol. **138**, 368–377 (2015).
[37]A. Köhler, A. Rasch, D. Pallarès, and F. Johnsson, "Experimental characterization of axial fuel mixing in fluidized beds by magnetic particle tracking," Powder Technol. **316**, 492–499 (2017).
[38]X. Tao, X. Tu, and H. Wu, "A new development in magnetic particle tracking technology and its application in a sheared dense granular flow," Rev. Sci. Instrum. **90**(6), 065116 (2019).
[39]H. Zhang, C. Wei, M. Zhao, Q. Liu, and H. Wu, "A novel convolutional neural network model to remove muscle artifacts from EEG," in *ICASSP 2021—2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2021).
[40]A. Rasti-Meymandi and A. Ghaffari, "A deep learning-based framework for ECG signal denoising based on stacked cardiac cycle tensor," Biomed. Signal Process. Control **71**, 103275 (2022).
[41]P. Xiong, H. Wang, M. Liu, S. Zhou, Z. Hou, and X. Liu, "ECG signal enhancement based on improved denoising auto-encoder," Eng. Appl. Artif. Intell. **52**, 194–202 (2016).
[42]J. Rock, M. Toth, E. Messner, P. Meissner, and F. Pernkopf, "Complex signal denoising and interference mitigation for automotive radar using convolutional neural networks," in *2019 22th International Conference on Information Fusion (FUSION)* (IEEE, 2019).
[43]C. Weng, D. Yu, S. Watanabe, and B.-H. F. Juang, "Recurrent deep neural networks for robust speech recognition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2014).
[44]P. Huang, M. Kim, M. A. Hasegawa-Johnson, and P. Smaragdis, "Singing-voice separation from monaural recordings using deep recurrent neural networks," in *International Society for Music Information Retrieval Conference* (IEEE, 2014).
[45]S. Poungponsri and X.-H. Yu, "Electrocardiogram (ECG) signal modeling and noise reduction using wavelet neural networks," in *2009 IEEE International Conference on Automation and Logistics* (IEEE, 2009).
[46]C. T. C. Arsene, R. Hankins, and H. Yin, "Deep learning models for denoising ECG signals," in *2019 27th European Signal Processing Conference (EUSIPCO)* (IEEE, 2019).
[47]W. Zhu, S. M. Mousavi, and G. C. Beroza, "Seismic signal denoising and decomposition using deep neural networks," IEEE Trans. Geosci. Remote Sens. **57**(11), 9476–9488 (2019).
[48]A. Fuchs, J. Rock, M. Toth, P. Meissner, and F. Pernkopf, "Complex-valued convolutional neural networks for enhanced radar signal denoising and interference mitigation," in *2021 IEEE Radar Conference (RadarConf21)* (IEEE, 2021).
[49]K. Antczak, "Deep recurrent neural networks for ECG signal denoising," arXiv:1807.11551 (2018).
[50]W. Wang, B. Li, and H. Wang, "A novel end-to-end network based on a bidirectional GRU and a self-attention mechanism for denoising of Electroencephalography signals," Neuroscience **505**, 10–20 (2022).
[51]Z. Zhao, Z. Wu, Y. Zheng, and P. Ma, "Recurrent neural networks for atmospheric noise removal from InSAR time series with missing values," ISPRS J. Photogramm. Remote Sens. **180**, 227–237 (2021).
[52]K. Tan and D. Wang, "A convolutional recurrent neural network for real-time speech enhancement," in Proceedings of Interspeech, Hyderabad, India, 2–6 September 2018, pp. 3229–3233.
[53]L. Sun, J. Du, L.-R. Dai, and C.-H. Lee, "Multiple-target deep learning for LSTM-RNN based speech enhancement," in *2017 Hands-Free Speech Communications and Microphone Arrays (HSCMA)* (IEEE, 2017), pp. 136–140.
[54]K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," Neural Networks **3**(5), 551–560 (1990).
[55]K. Hornik, "Approximation capabilities of multilayer feedforward networks," Neural Networks **4**(2), 251–257 (1991).
[56]T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation* (Prentice Hall, 2000).
[57]R. J. Williams and J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," Neural Comput. **2**(4), 490–501 (1990).
[58]I. Sutskever, "*Training recurrent neural networks*," Ph.D. dissertation, University of Toronto, 2013.
[59]S. Jeong, M. Ferguson, R. Hou, J. P. Lynch, H. Sohn, and K. H. Law, "Sensor data reconstruction using bidirectional recurrent neural network with application to bridge monitoring," Adv. Eng. Inf. **42**, 100991 (2019).
[60]W. Choi and S. Lee, "Performance evaluation of deep learning architectures for load and temperature forecasting under dataset size constraints and seasonality," Energy Build. **288**, 113027 (2023).
[61]Z. Niu, G. Zhong, G. Yue, L.-N. Wang, H. Yu, X. Ling, and J. Dong, "Recurrent attention unit: A new gated recurrent unit for long-term memory of important parts in sequential data," Neurocomputing **517**, 1–9 (2023).
[62]G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou, "Minimal gated unit for recurrent neural networks," Int. J. Autom. Comput. **13**(3), 226–234 (2016).