Highlights

DeepTrackStat: an End-to-End Deep Learning Framework for Extraction of Motion Statistics from Videos of Particles

Marc Berghouse,Rishi Parashar

- We propose DeepTrackStat, a novel end-to-end framework composed of multiple convolutional neural networks and vision transformers with class-based ensembling that generates fast and accurate predictions for speed, velocity component (V_x & V_y), and turn angle distributions of particles.
- DeepTrackStat was specifically developed to generate motion statistics of particles from the types of videos found in microfluidics and microscopy studies.
- DeepTrackStat outperforms a variety of classical particle tracking algorithms in the task of motion statistic prediction.
- DeepTrackStat performance is especially strong at high speeds compared to classical particle tracking algorithms.

DeepTrackStat: an End-to-End Deep Learning Framework for Extraction of Motion Statistics from Videos of Particles

Marc Berghouse[a,b,*], Rishi Parashar[b]

[a]University of Nevada, Reno, 1664 N Virginia St, Reno, 89957, NV, United States
[b]Desert Research Institute, 2215 Raggio Pkwy, Reno, 89512, NV, United States

ARTICLE INFO

Keywords:
Particle Tracking
Deep Learning
Computer Vision
Bioimage Analysis
Porous Media
Object Tracking
Motion Prediction

ABSTRACT

Particle tracking (PT) is a mature area of research that traditionally has used Gaussian filtering and nearest neighbors-based algorithms to detect and link features in a sequence of images. PT shares many similarities with the general task of object tracking, although it is specifically designed for tracking objects that are usually small and have a distinct shape shared amongst all particles in images with little to no background. Although object tracking is also a mature area of research, transferring the computer vision techniques from general object tracking to PT presents significant challenges due to the sparseness of particles and high resolution of PT videos. To remedy these issues, we present DeepTrackStat (DTS), a novel class-based deep learning framework for the engineering application of extracting motion statistics from videos of particles. DTS is able to bypass the tracking process entirely and generate accurate statistics on speed, velocity components, and turn angle for a wide range of PT scenarios including trajectories derived from Brownian motion, Poiseuille flow, and porous media flow. The model is robust to large variations in particle size, shape, brightness, speed, density, and signal to noise ratio, and can reduce the time required to obtain the target statistics from videos of moving particles by 6x (when compared with classical methods). In addition, we show that DTS' performance is comparable to a state-of-the-art (SOTA) method for a variety of simulated trajectories and experimental datasets of motile bacteria dispersing in porous media under a range of flow conditions.

1. Introduction

At a basic level, Particle Tracking (PT) is a set of algorithms used to detect bright spots and determine their trajectories across multiple frames of a video. PT can be considered a subset of the object tracking task, and can be applied to any video data with moving objects, but it is especially relevant for tracking small, spherical particles. This includes phenomena such as bacterial dispersion and transport in porous media [1-9], cellular diffusion [10, 11], biofilm formation [12], chemotaxis [13-15], viral transport [16], and colloid filtration [17]. These applications of PT are highly dependent on algorithm's capability to accurately extract particles' motion statistics. Reliable estimates of particles' speed, velocity components, and turning angles measured along the length of a trajectory are essential ingredients for developing predictive models of transport of viewable substances in natural and engineered porous media [18, 19]. Inaccurate or incomplete measurement of motion statistics leads to erroneous estimations of bulk transport metrics. For example, when PT algorithms tend to miss fast-moving particles, it can lead to an underestimation of the mean square displacement of the spreading plume of particles [20]. Microfluidic studies paired with advance imaging techniques are often conducted to first record videos of dispersing particles which are then analyzed by PT algorithms to characterize particles' motion behavior in controlled settings [21, 22, 23]. The motion is then mathematically and/or numerically upscaled to provide estimates of bulk transport properties at range of scales relevant to engineering applications [24, 25].

Most PT frameworks consist of detection, linking, and filtering stages [26]. The detection stage generally uses a Gaussian filter to filter and normalize an image, thus revealing local maxima that correspond to the centroids of spherical objects of a certain diameter. Many strategies have been developed to improve upon this general detection method [27, 28], and the most state of the art detection algorithms use convolutional neural networks (CNNs) to improve particle recognition [29-31]. The linking stage consists of connecting the detected bright spots across multiple

*Corresponding author
mberghouse@unr.edu (M. Berghouse); rishi@dri.edu (R. Parashar)
ORCID(s): 0000-0002-2278-1392 (M. Berghouse); 0000-0002-9733-7309 (R. Parashar)

frames in time to form the most probable trajectory for each particle, and is generally where most PT algorithms differ from each other. For linking of fast and dense particles, TrackMate (TM) [32] has been shown to be one of the best performing PT algorithms [20, 26]. However, TM is also slow, and it may take significant domain knowledge and experimentation to achieve accurate tracking results. Unlike detection, very few deep learning (DL)-based methods have been developed for the linking stage. The only algorithm that has been presented as an end-to-end DL-based framework for linking particle trajectories is MAGIK [33], which uses a graph neural network to capture the spatiotemporal relationships present in PT coordinate data. Although this algorithm boasts strong performance for the tested scenarios, it is untested in scenarios of high particle speed and density, and requires a large amount of VRAM (a coordinates shape of 100 frames by 1000 particles requires at least more than 24GB). Furthermore, MAGIK still requires another algorithm to perform the detection stage, and then the coordinate data must be converted to node-edge format for model use (which is not a trivial step), meaning it does not improve the ease, speed, or accuracy of the overall particle tracking process. MAGIK can also be thought of as one of the few models that bridge the gap between the fields of particle tracking and object tracking. Object tracking is a mature field within the domain of computer vision that uses CNNs and vision transformers (ViTs) to primarily track people and cars [34, 35]. Although object tracking methods are robust in their task-relevant performance, few have been trained to track the kinds of objects generally found in particle tracking experiments.

In an effort to bridge the gap between the particle tracking and object tracking domains and improve upon the extraction of motion statistics from PT data, we propose DeepTrackStat (DTS), a novel end-to-end DL-based framework. Specifically, our model offers the ability to predict speed, velocity (V_x and V_y) and turn angle distributions from a raw image sequence input. DTS is designed to be as general as possible, meaning it can accurately predict statistics from a variety of particle shapes, sizes, brightness, density, speed, and signal to noise ratio, and a variety of trajectory motion types such as dispersive, straight, and Brownian. DTS is a two-stage system that consists of a speed classifier (SC) and statistics-specific models (SSMs). An input image sequence is first classified according to mean speed, then based on this classification, different ensembles of models are used to generate the final predictions for each set of statistics. We show that our proposed class-based ensembling method largely outperforms a simple ensembling method and multiple classical PT algorithms. Furthermore, our method significantly outperforms three popular classical PT algorithms and slightly outperforms TM (a SOTA classical algorithm) over the whole test set, and it significantly outperforms TM when only analyzing videos with high-speed particles. Finally, we find that DTS can offer significant time savings for the extraction of motion statistics compared to classical PT algorithms as it's measured to be around 6x faster than TM.

2. Data and methods

2.1. Simulated data

DTS was developed with the goal of extracting motion statistics from a wide range of videos of particles. One of the challenges of this task is that there are no common benchmarks that currently exist for the specific task of measuring particle motion extraction capabilities. Thus, we developed a novel dataset containing a wide variety of particle tracking cases to properly test DTS. To create a highly general model, we generated over 2000 simulations of moving particles that were used for training. All simulations were 40 frames long, 2000 by 2000 pixels, and 1 channel (grayscale). The simulations differed in two primary ways - image and motion properties - in order to train the model on a wide variety of spatiotemporal conditions. Samples of the types of imagery and the distributions of motion statistics generated from our simulations can be seen in Figure 1. The image properties we varied were particle shape, size, density, seeding location, brightness, and signal to noise ratio (SNR). To change the SNR of the simulations, we used varying combinations of Gaussian, speckled, and salt and pepper noise. The motion properties we varied were particle speeds and pathlines. The pathlines were either generated randomly (to represent Brownian motion) or from flow fields representative of flow in porous media, straight advective flow, or Poiseuille flow. The porous media pathlines were largely generated from flow fields of heterogeneous geometries (created both in OpenFOAM [36] and via the Lattice Boltzman Method). A small percentage of the simulated data represents that of a homogeneous porous geometry consisting of a staggered array of cylinders. This type of homogeneous geometry has been used for particle tracking studies in the fields of bacterial motility and deterministic lateral displacement [2, 37]. Likewise, Brownian motion, and Poiseuille and heterogeneous porous media flows represent a large range of the motion observed in the body of cell microscopy data. Thus, our image and particle motion varieties aim to capture the most commonly encountered types of video data for both microfluidics and general microscopy experiments. The simulated training and testing set

don't significantly differ. Different parameters (particle speed, density, SNR, shape and seeding location) were used to generate the testing simulations than the training simulations, but the range of distributions of variables from the testing simulations generally falls within the range of distributions of variables from the training simulations (Fig. 1). In an effort to display the robustness of DTS, the simulated test set aims to replicate most of the variation in the simulated training set.

The aim of DTS is to predict speed, velocity component, and turn angle distributions directly from videos of moving particles. We chose to focus on the prediction of these motion statistics because they are important baseline measurements to understand the advective-diffusive transport of particles such as colloids and bacteria. However, we believe that frameworks like DTS can be extended to other statistics, so our work also serves as a proof of concept for researchers who may be looking for more task-relevant statistics such as dispersion coefficients or mean square displacements. For speed, we predict the magnitude of the ensemble velocity of the particles in pixels per frame as $S = \sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2}/\Delta t$, where t represents time (in frames) and $\Delta t = 1$. For velocity, we predict the ensemble x and y velocity components (V_x and V_y) in pixels per frame. For turn angle, we predict the relative change in direction of the ensemble of particles between two successive frames as $\alpha_t = \arctan(\frac{y_{t+2}-y_{t+1}}{x_{t+2}-x_{t+1}}) - \arctan(\frac{y_{t+1}-y_t}{x_{t+1}-x_t})$. A low average turn angle corresponds to particles that primarily move straight, and a high average turn angle corresponds to particles that have a high probability of changing directions between frames.

2.2. Experimental data

In addition to our simulated imagery/trajectories, we also test the performance of DTS on experimental videos. These videos are from microfluidics experiments of motile bacteria in porous and open media. Specifically, we use videos of *Acidovorax* [38], *Geobacter* [39], *Paenibacillus* [40], and *Shewanella* [41] moving through structures with varying levels of porosity ($\phi = 0$, $\phi = 0.42$, and $\phi = 0.6$) and at varying flow rates (0, 1, and 5 $\mu l/h$). Because this is an experimental dataset, there is no ground truth. Thus, we use the results from TM as a relative ground truth to gauge the performance of DTS. The experimental videos range from 200 to 3000 frames, and are 2048 by 2048 pixels.

2.3. Model development

We present a novel end-to-end framework that consists of two stages: the speed classifier (SC) and the statistics-specific models (SSMs). Previous studies have presented similar class-based ensemble methods for various tasks [42, 43, 44]. However, our model is novel in its combination of architectures used, the use of a speed classifier to improve predictions of motion statistics, and the task of extracting information from videos of moving particles. The framework splits an input video into 40-frame chunks and averages the motion statistics predictions across all chunks, meaning the model can process any grayscale video input with at least 40 frames. The SC uses an ensemble of convolutional neural networks (CNNs) and vision transformers (ViTs) to classify the input into one of five classes based on speed. All models used in the ensemble can be seen in the publicly-available testing script, but the models that carry the most weight in the SC ensemble are VoloD1-384 [45], Pyramid Vision Transformer V2-b1 [46], RegnetX-032 [47], and VoloD3-448 [45], which were chosen for their high single-model performance. CNNs are well known to be able to capture spatial features within the images of a video [48], but may have trouble learning the temporal relationships in the data [49]. Thus, we also use ViTs, which are especially suited to learn features in sequences of images [50], and have shown high performance on video classification tasks [51]. The speed classifier tries to predict ranges of mean particle speeds. Specifically, class 1 corresponds to a mean speed of 0-2 pixels/frame, class 2 is 2-5 pixels/frame, class 3 is 5-10 pixels/frame, class 4 is 10-18 pixels/frame, and class 5 corresponds to a mean particle speed of greater than 18 pixels/frame.

The second stage of the DTS framework, the SSMs, consists of a variety of ensemble models for each statistic and each class. The specific models used in each ensemble were determined by their single-model performance. DTS outputs a sorted 500-length vector of probable values for particle speeds, turn angles, and velocity components (V_x and V_y). Through this framework, the speed classifier has a large impact on the final results, with each specific SSM only slighlty shifting the value of the outputs. The ensemble weights for the SC and SSMs were determined through calibration of 50% of the simulated test data. In addition to calibration, we used simple boolean logic to improve DTS' performance on Brownian trajectories and trajectories that are relatively straight, but this feature has to be manually specified by the user. If the user knows a particular video contains primarily Brownian trajectories and sets this flag, then DTS will ensure that the output for V_x has a mean value of 0. Likewise, if the user observes that their video contains a large majority of particles that move straight, DTS will use a different ensemble for the turn angle distribution. Although DTS still has comparable performance to TrackMate without the use of these special flags, their use improves results

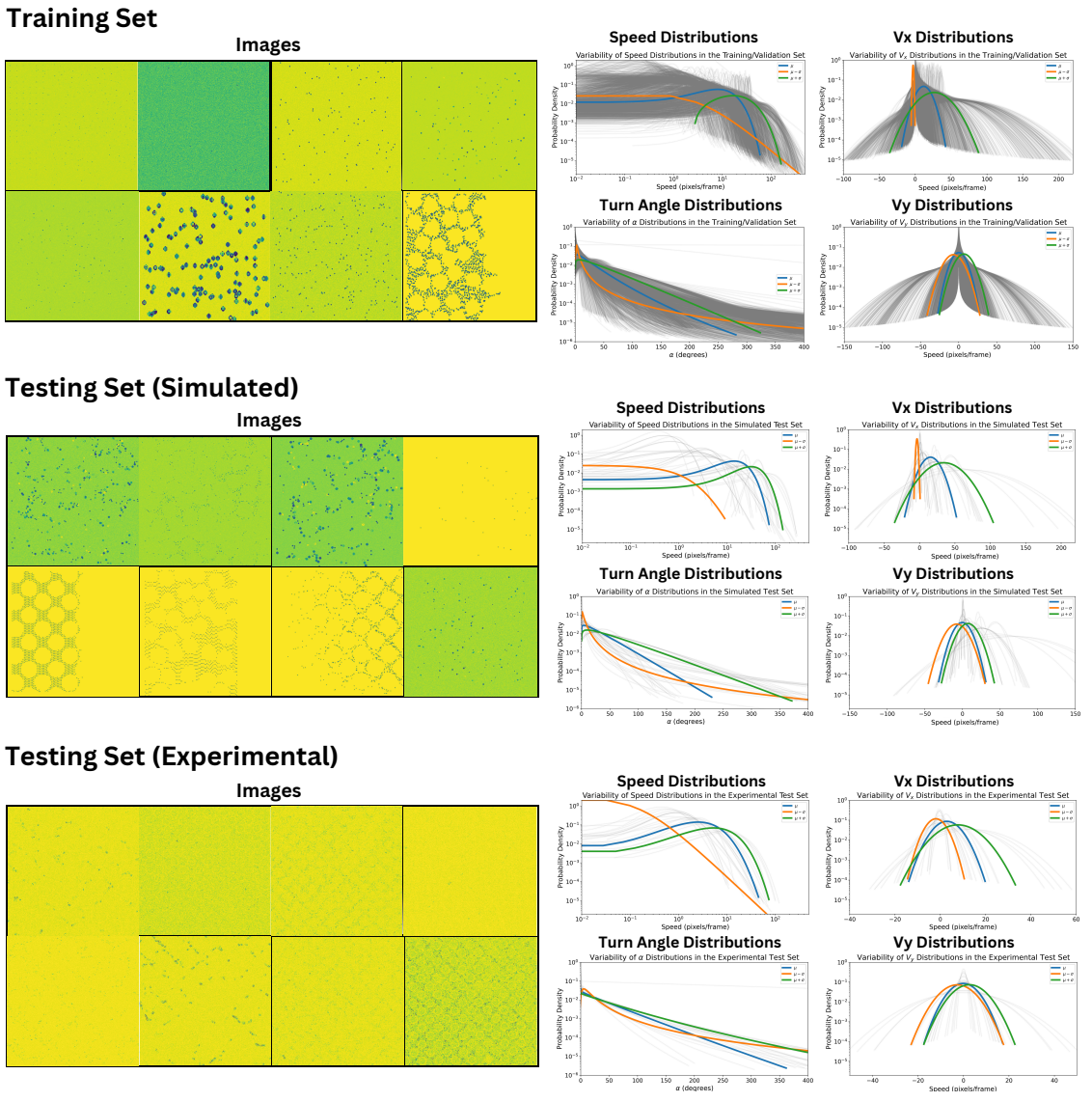

Figure 1: Images and motion statistic distributions for the training, simulated test, and experimental test data. The images are grayscale (1 channel) and are presented here as false-color images to highlight differences in brightness. The wide variety of images (in terms of particle density, shape, size, and image noise) illustrate the range of inputs that DTS is able to extract accurate predictions from. The blue distribution is the mean of the respective set (training, simulated testing, experimental testing), the orange is the mean minus one standard deviation, and the green is the mean plus one standard deviation. The grey distributions show the full range of variability for the respective set. The distributions for the simulated and experimental test datasets are mostly captured in the training dataset.

for the specific cases of Brownian and straight particles. For all results discussed in the paper, both flags were used to improve performance for these trajectory types.

The speed SSM primarily consists of a 4xVoloD1-224 patch model, VoloD3-448, VoloD1-384, and VoloD2-384. The patch model takes in a downsampled video input (448x448 pixels) and splits it into four 224x224 patches. Each of these patches is then fed into a VoloD1-224 model with an output size of [B,500]. The outputs from each VoloD1-224 model are then concatenated and fed into a fully connected layer to get the final desired output shape of 500. For all other models, the final classification layer is simply replaced to get an output shape of 500. After each model generates its outputs, the class-based ensemble weights are used to generate the final model outputs. The exact models and ensemble

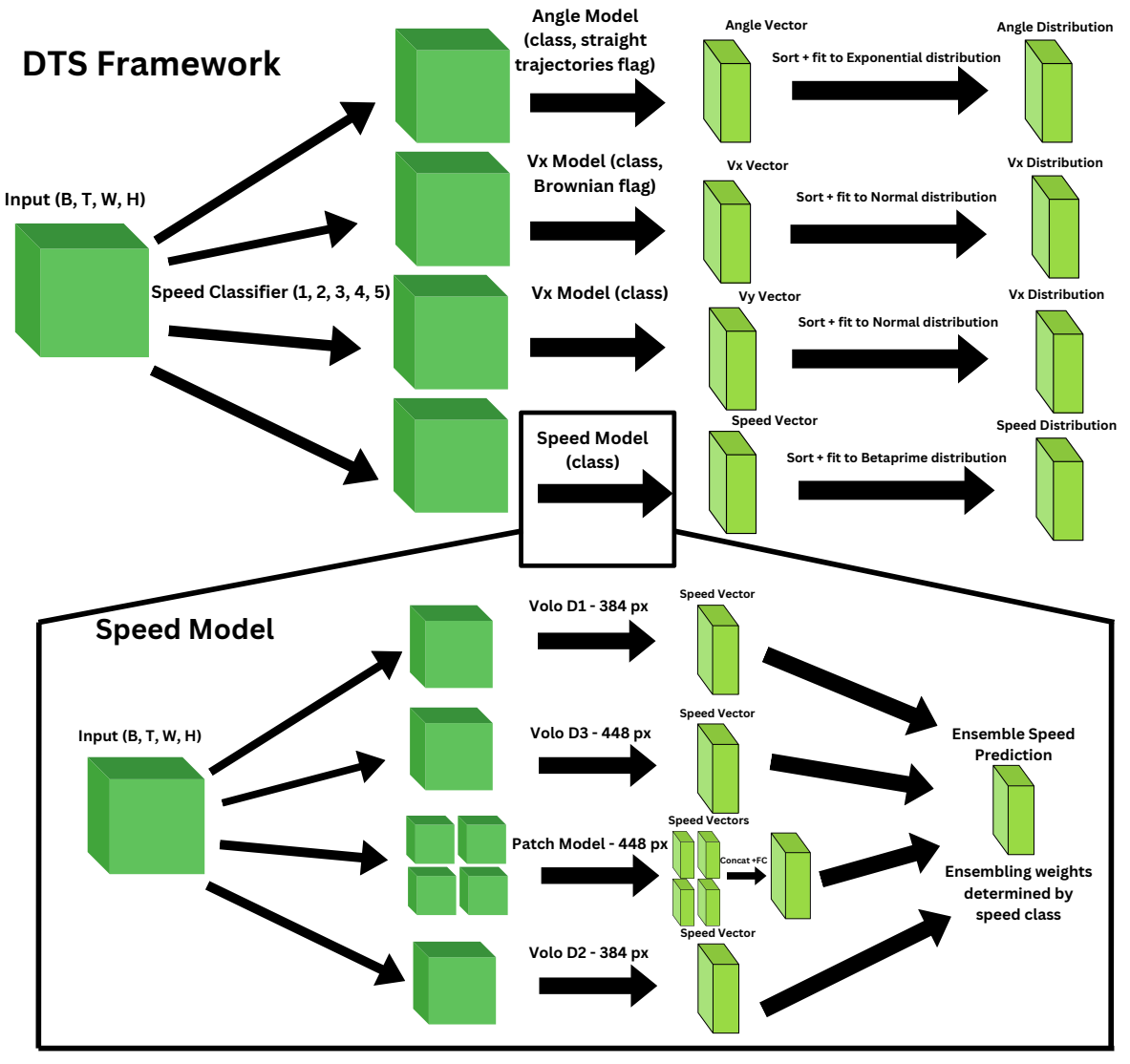

Figure 2: Overall framework for the proposed model (DTS). The model accepts a grayscale video as input (T must be 40, W and H must be equal) and first send it through the speed classifier (SC), which is an ensemble model used to classify the video of particles into 5 speed categories. The input is then sequentially sent to statistics-specific models (SSMs), which are each individually trained for their specific prediction task. Based on the output of the speed classifier, each SSM uses a different conditional ensemble model to generate the predictions. The outputs of DTS are the raw 500-length vectors of values for each statistic, and the respective distribution for each statistic.

weights used in each stage are given in Supplementary Figure 1. All base models were constructed with the PyTorch Image Models (TIMM) repository [52], meaning the final classification layer was changed via the "num_classes" flag. All models used the default pre-trained weights from TIMM (model-specific, but mostly ImageNet [53]). The turn angle, V_x, and V_y SSMs are constructed from similar ensembles, the details of which can be viewed in the code or Supplementary Figure 1.

2.4. Training and testing process

We used 1923 simulations for training and 481 simulations for validation, which was used to reduce overfitting during training via early stopping. The simulated test set contains 43 simulations and the experimental test set contains 23 pairs of images and trajectories. For TIMM models, the set dropout rate applies increasingly larger amounts of

dropout in the transition layers of the model, with the final transition layer having the set amount of dropout. All patch-based models were trained with dropout of 0.3, and all other models were trained with dropout of 0.4. All models were trained with the AdamW optimizer at a learning rate (lr) of between 1.2e-5 and 2e-4. The speed classifier was trained with an lr of 2e-4, the speed SSM was trained with an lr of 1.2e-5, the V_y SSM was trained with an lr of 1e-4, the V_x SSM was trained with an lr of 1.2e-4, and the α SSM was trained with an lr of 6e-5. Hyperparameter tuning (dropout, learning rate, and number of classes) was done in a two step process. We tuned the hyperparameters automatically via Optuna [54] for a few models for each statistic, then used the best range of learning rates to manually test a few sets of hyperparameters for each of the other models. All speed classifier models were trained for 400 epochs, speed prediction models were trained for 90 epochs, α models were trained for 300 epochs, V_y models were trained for 95 epochs, and V_x models were trained for 150 epochs. All "224" (i.e. models that take in an input of 224x224) models were trained with a batch size of 32, all "384" models were trained with a batch size of 16, and all 448 models were trained with a batch size of 8. All training and testing for DTS, and all PT experiments, were performed on a CUDA-capable computer with an Nvidia 4090 GPU, Intel i9-14900KF CPU, and 96GB of RAM.

In this paper we compare the performance of DTS to four other algorithms (TrackMate, Trackpy [55], TracTrac [56], and LapTrack [57]). For TrackMate, we used the Kalman filter linking algorithm for trajectories with directed motion and the LAP linking algorithm for trajectories with Brownian motion. Aggregate motion statistics for each classical PT algorithm (and the ground truth) were computed by ensemble averaging methods over all trajectories and frames. Essentially, each tracker outputs a csv of the trajectories that are sorted and looped through to calculate ensemble statistics. Statistics for DTS are calculated as the ensemble of all outputs from a single video.

Each model was calibrated to achieve the best results on the testing set. For TM, Trackpy, TracTrac and LapTrack, calibration was performed through a cycle of visual and statistical analysis to inform the adjustment of tracking parameters. For DTS, calibration entailed adjusting the class-based ensemble weights to achieve the best possible results on 50% of the testing data. The point of the calibration step is to simulate the scenario of using DTS to extract motion statistics from multiple videos. Given the rigorous testing in scientific literature that classical PT algorithms have gone through, it is reasonable to first use a classical PT framework (such as TM) to produce motion statistics in order to verify the accuracy of DTS for a particular dataset. If there are any significant discrepancies between DTS and TM, the ensemble weights of DTS can then be adjusted to match TM (or any other SOTA tracking algorithm), ensuring accurate statistics for the particular data being analyzed. In addition to our calibrated results, we also provide uncalibrated results for DTS. In the uncalibrated version of DTS, the ensemble weights were determined through optimizing predictions of the validation set.

3. Results and discussion

3.1. Decreased run time

One of the primary advantages of DTS over classical PT algorithms (such as TM) is the reduced computation time for generation of motion statistics. Particle trajectory analysis often requires many imaging trials at high resolution, meaning the time required to extract motion statistics is an important concern. For one of our experimental videos with dense particles that contains 2480 frames, TM takes 2.5 minutes for loading the images into ImageJ [58], 5 minutes to perform the detection step, 4 minutes to perform the linking step, 0.5 minutes to filter and export the trajectories, and 0.5 minutes to calculate the statistics, which means the TM framework in total takes 12.5 minutes to extract statistics from the video data. This is assuming that OOM errors aren't encountered (a 2480 frame video of 2048x2048 resolution with >1000 particles in each frame will cause TM to crash) and that the tracking parameters used on the first try are optimal, which is unlikely for anyone besides an expert in the field. Even for someone experienced with PT codes, a 2480 frame video with over 1000 particles per frame will likely require 30 minutes to get good results. In stark contrast, DTS only takes 2 minutes to make its predictions for the same video, and requires significantly less domain knowledge to get accurate predictions of motion statistics. Additionally, DTS always takes two minutes for a 2480 frame video of 2048x2048 resolution, whereas the time required to generate statistics via classical PT methods significantly depends on the number of trajectories. For videos with very few particles, DTS may not save much time, but for videos with a large number of particles (>1000), DTS will save a significant amount of time.

3.2. Ablation experiments

In order to show the benefit of our proposed model structure, we performed ablation experiments for each statistic. We report the mean average error (MAE) plus or minus the standard deviation of the mean value of each statistic across

Table 1
Ablation experiments on the simulated test set (n=43 samples). Scores are reported as the MAE for all samples in the test set plus or minus one standard deviation of the MAE between all samples of the test set. The metrics are non-negative with a large positive skew that often results in a standard deviation greater than the mean. For all proceeding tables, results are reported as the mean error with a 10th-90th range to clear any potential confusion. Here we give the results for the top 4 sets of single models, a simple ensemble of MS-1, MS-2, and MS-3, and a 4-var model that uses a single model (VoloD1-384) to predict all statistics at once, and DTS (the proposed framework). The best performing single models for speed are VoloD3-448 (MS-1 & MS-3) and the 4xVoloD1-224 patch model (MS-2 & MS-4). For V_x the best performing models are VoloD1-384 (MS-1), RegnetX-016 (MS-2), VoloD3-448 (MS-3), and RegnetX-032 (MS-4). For V_y the best performing models are VoloD4-448 (MS-1), VoloD3-448 (MS-2), VoloD1-384 (MS-3) and VoloD1-224 (MS-4). For turn angle the best performing models are VoloD3-448 (MS-1 & MS-3) and VoloD1-384 (MS-2 & MS-4). The class-based model significantly ($p < 0.05$) outperforms the simple ensemble for the speed predictions and slightly outperforms the ensemble in all other metrics.

Stat	MS-1	MS-2	MS-3	MS-4	Ensemble	4-Var Model	DTS
S	3.7 ± 5.0	4.2 ± 6.1	4.3 ± 6.0	9.7 ± 15	5.2 ± 7.8	10 ± 12	$\mathbf{2.8 \pm 3.3}$
V_x	5.7 ± 4.6	5.8 ± 4.8	6.0 ± 5.5	6.8 ± 6.4	5.7 ± 4.8	9.9 ± 12	5.1 ± 9.0
V_y	0.8 ± 0.7	0.9 ± 0.6	0.9 ± 0.9	1.0 ± 0.9	0.8 ± 0.8	6.5 ± 7.2	0.5 ± 0.5
α	3.4 ± 2.9	3.5 ± 2.7	3.7 ± 3.9	3.9 ± 4.1	3.0 ± 2.7	5.4 ± 5.3	2.5 ± 2.1

all simulated test data for the four best sets of single models, a simple ensemble of the best models, a single model that outputs all four variables at once, and our proposed class-based model (Table 1). The set of single models represent the top 4 single models for each variable. For example, MS-1 gives the results for a VoloD3-448 model used to calculate speed, a VoloD1-384 model used to calculate V_x, a VoloD4-448 model used to calculate V_y, and a VoloD3-448 model used to calculate α. In the 4-Var model, there are four separate VoloD1-224 models that generate the feature maps for each variable, then these four feature maps are concatenated and passed through a linear layer to generate the final output of shape [B, 500, 4]. In this case, we see performance is dramatically worse than that of the set of single models, the simple ensemble and DTS, which indicates the need to develop an ensemble of single models.

For all statistics besides V_x, we find that the class-based model largely outperforms the best sets of single models. Furthermore, DTS significantly ($p < 0.05$) outperforms a simple ensemble of the best single models for the speed and turn angle prediction tasks. In the case of V_x, although the MAE is greater for DTS than for model 1, other metrics (RMSE and W1) indicate that DTS has a better overall fit to the ground truth data. Thus, we illustrate that a class-based ensembling method can lead to significant performance increases for the task of predicting motion statistics from videos of particles. Furthermore, the class-based method contains a large number of parameters that can be manually fine-tuned (such as the Brownian and straight motion flags, and easily modifiable weights for the SC and SSMs), which allows for more precise calibration depending on the range of videos that need to be analyzed.

The full DTS framework has around 2 billion (1,994,005,238) parameters. The single models used in this study have between 8 and 200 million parameters. Using all the best single models, the full prediction framework would have around 330 million parameters. The ensemble framework uses 3 models for the prediction of each statistic, which gives it around 1 billion parameters. The 4-var model is just a single VoloD1-384 model for all statistics, so this would only have 26 million parameters. Although the 4-Var model performs significantly worse than all others, this framework could be advantageous for situations where rapid predictions and low computational cost are required or preferred over accuracy.

3.3. Simulated test set

To ensure that DTS can handle a wide variety of simulated data, we included test simulations that had large variations in image and trajectory properties (Fig. 1). Our results indicate that the performance of DTS can match that of TM across this wide variety of simulations (Table 2). Specifically, DTS significantly outperforms TM in 2 out of 3 of the speed and angle metrics. Furthermore, DTS vastly outperforms other well-known PT methods such as Trackpy (Table 3), TracTrac (Table 4), and Laptrack (Table 5). In addition, we tried to compare the performance of DTS with a SOTA optical flow method [59], but determined that the model would require fine-tuning to perform the desired task (Supplementary Figure 2).

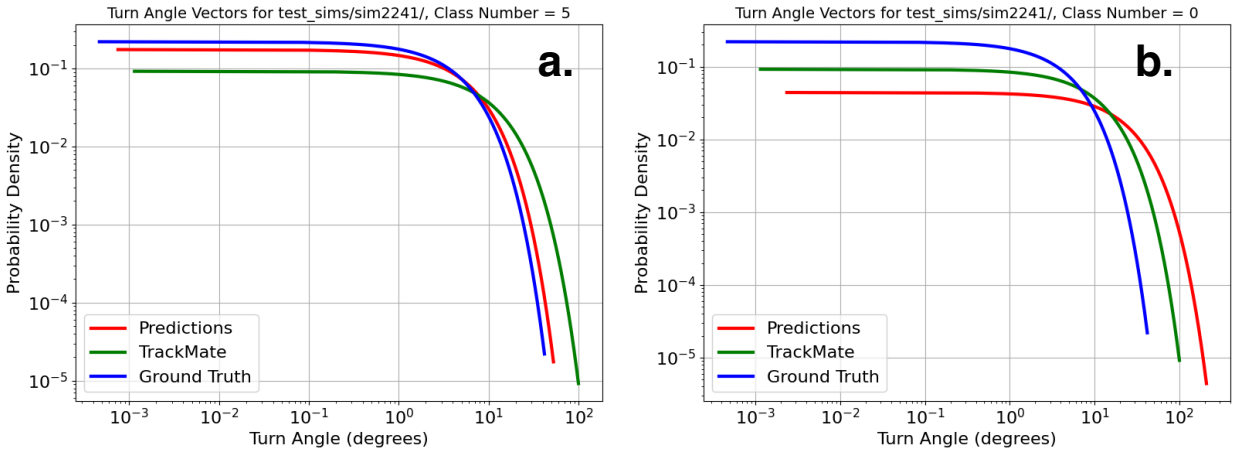

Figure 3: Impact on turn angle predictions of using the "straight trajectories flag" for a simulation of straight-moving particles. (a) Predictions with the flag on. (b) Predictions with the flag off. DTS is unable to accurately predict the turn angle distribution without manual help, illustrating the benefit of the flag, and the need to visually inspect the inputs before using DTS.

Table 2
Results from the simulated test set (n=43 samples) for each statistic comparing DTS and TM. The mean of each statistic is given along with the 90th-10th percentile error range. For both frameworks, we give the MAE, RMSE, and 1-Wasserstein distance (W1) for each statistic (relative to the ground truth). Statistically significantly better performances are bolded.

	DeepTrackStat			TrackMate		
Stat	MAE	RMSE	W1	MAE	RMSE	W1
Speed	**2.80** [**.116**, **7.18**]	**5.37** [**.839**, **11.6**]	.026 [.0005, .082]	7.41 [.030, 28.0]	11.1 [.610, 31.8]	.016 [.0015, .053]
V_x	5.05 [.047, 15.7]	8.48 [.828, 20.6]	.076 [.0013, .113]	7.95 [.004, .342]	12.0 [.712, 34.2]	.087 [.0010, .154]
V_y	.547 [.012, 1.27]	4.63 [.609, 12.9]	.026 [.0005, .108]	**.291** [**.005**, **.846**]	4.95 [.231, 11.9]	.021 [.0004, .090]
α	**2.51** [**.529**, **5.23**]	**5.41** [**1.57**, **9.63**]	.002 [.00002, .002]	5.49 [.987, 9.59]	10.5 [3.90, 20.8]	.002 [.00017, .004]

Complementary to our calibrated results for DTS, we also present results for an uncalibrated framework (Table 6). While the uncalibrated framework doesn't perform as well as the calibrated framework (Table 5), it does indicate that DTS can be used out-of-the-box to predict motion statistics with much greater accuracy than the calibrated predictions of Trackpy (Table 2), TracTrac (Table 3), and Laptrack (Table 4), and slightly better accuracy than the calibrated predictions of TrackMate (Table 5). Additionally, these results show that the class-based framework has a clear advantage over simple ensembling for speed and angle predictions(Table 1).

DTS shows especially strong performance for simulations with high particle speeds (Table 7). We define high-speed simulations as having a mean ensemble speed of greater than 25 pixels/frame. For our 12 test simulations that meet this criteria, DTS dramatically outperforms TM, showing improvement in every metric for the speed, V_x, and turn angle statistics. Once again, the speed and angle predictions stand out, with both the MAE and RMSE showing a statistically significant improvement from TM. For traditional particle tracking methods (ie not based in deep learning methods) such as TM, the quality of the extracted trajectories is mainly determined by particle spacing displacement ratio (PSDR), which is the ratio of the average spacing between any two particles and the average speed of a particle. Since TM, and most other classical PT methods, are all roughly based on some kind of nearest neighbors approach, the lower the PSDR, the harder it is for them to accurately track the particles. At high particle speeds, the PSDR is low, so TM is unable to extract accurate trajectories. DTS, since it is not based on any kind of nearest neighbors algorithm and does not actually perform tracking, doesn't suffer from this issue. For speed, The MAE for DTS is about 5x less than that of TM, showing that DTS has a clear application for improving the accuracy of speed predictions for videos of high-speed particles.

Table 3
Results from the simulated test set (n=43 samples) for each statistic for Trackpy. The mean of each statistic is given along with the 90th-10th percentile error range. The performance of Trackpy is considerably worse than that of DTS or TM.

Stat	MAE	RMSE	W1
S	8.44 [.306, 26.0]	10.8 [.508, 30.4]	.038 [.0008, .142]
V_x	8.87 [.392, 27.2]	11.6 [.662, 32.9]	.032 [.0015, .049]
V_y	3.42 [.174, 13.5]	4.87 [.364, 16.9]	.037 [.0002, .172]
α	8.42 [1.06, 20.4]	13.2 [1.77, 28.8]	.003 [.0002, .005]

Table 4
Results from the simulated test set (n=43 samples) for each statistic for TracTrac. The mean of each statistic is given along with the 90th-10th percentile error range. TracTrac is the lowest-performing PT method that was tested in this study.

Stat	MAE	RMSE	W1
S	12.0 [.029, 32.2]	14.2 [.068, 32.9]	.032 [.0017, .046]
V_x	12.8 [.045, 39.9]	14.8 [.156, 45.2]	.026 [.0022, .031]
V_y	4.06 [.023, 13.0]	5.32 [.179, 16.4]	.032 [.0015, .075]
α	14.5 [.548, 42.3]	14.5 [.548, 42.3]	.002 [.00007, .006]

Table 5
Results from the simulated test set (n=43 samples) for each statistic for Laptrack. The mean of each statistic is given along with the 90th-10th percentile error range. The performance of Laptrack is considerably worse than that of DTS or TM.

Stat	MAE	RMSE	W1
S	8.14 [.533, 22.3]	11.1 [1.33, 25.7]	.035 [.0006, .085]
V_x	8.69 [.584, 22.7]	12.5 [3.20, 29.6]	.032 [.0015, .063]
V_y	3.63 [.500, 10.4]	6.29 [1.19, 13.3]	.041 [.0006, .171]
α	10.4 [.964, 20.1]	15.5 [2.23, 29.0]	.003 [.0002, .005]

Table 6
Uncalibrated results from the simulated test set (n=43 samples) for each statistic for DTS. For these results, the ensemble weights and chosen models for DTS were determined via performance on the validation set, meaning these represent the general performance capabilities of DTS for completely unseen data. While the results aren't as strong as the calibrated ones, the errors in speed and angle prediction are still less than any other method tested in this paper.

Stat	MAE	RMSE	W1
S	3.23 [.25, 7.7]	5.32 [.30, 12.1]	.021 [.00046, .039]
V_x	5.20 [.03, 20.2]	8.62 [.91, 22.4]	.026 [.00104, .077]
V_y	.627 [.03, 1.8]	4.25 [.39, 10.4]	.025 [.00153, .081]
α	2.58 [.53, 5.2]	5.35 [1.3, 9.81]	.002 [.00002, .002]

In addition to our numerical performance comparison of DTS and TM, we also present a graphical performance comparison of the distributions of the simulated and experimental test sets (Fig. 4) for each statistic. The distributions obtained from the simulated test sets (Figs. 4a, 4b, 4c, and 4d) show that, on average, the distribution shapes obtained from DTS closely resemble the ground truth trajectories. Furthermore, for V_x, although the fit between distributions (as measured by W1) is usually closer to the ground truth for TM on a simulation-by-simulation basis, the predictions by DTS are much more accurate for the high-speed simulations. Thus, when looking at the distribution for all data, the predictions from DTS show much better alignment with the ground truth than TM does. This importantly shows that DTS has a lower chance of correctly predicting the true V_x distribution for individual videos, but a higher chance of predicting the true V_x distribution for a group of videos.

Table 7
Average results from a high-speed subset (n=12 samples) of the simulated test data. Statistically significantly better performances are bolded. In the case of high-speed PT data, DTS performs better than TM in every metric.

	DeepTrackStat			TrackMate		
Stat	MAE	RMSE	W1	MAE	RMSE	W1
Speed	**4.33 [1.25, 9.35]**	**7.93 [2.21, 15.5]**	.002 [.0003, .003]	22.8 [2.36, 46.4]	27.5 [9.31, 51.9]	.004 [.002, .006]
V_x	12.9 [.718, 35.6]	18.6 [7.31, 43.4]	.050 [.0060, .027]	24.7 [2.40, 51.9]	28.9 [8.58, 56.4]	.146 [.0540, .161]
V_y	.882 [.249, 1.38]	8.70 [3.06, 16.3]	.004 [.0003, .004]	.775 [.190, 1.41]	9.51 [3.46, 17.2]	.003 [.0004, .009]
α	**2.85 [.400, 5.36]**	**5.03 [1.24, 7.88]**	.002 [.00005, .002]	5.60 [2.71, 7.68]	13.5 [5.35, 20.2]	.003 [.0006, .002]

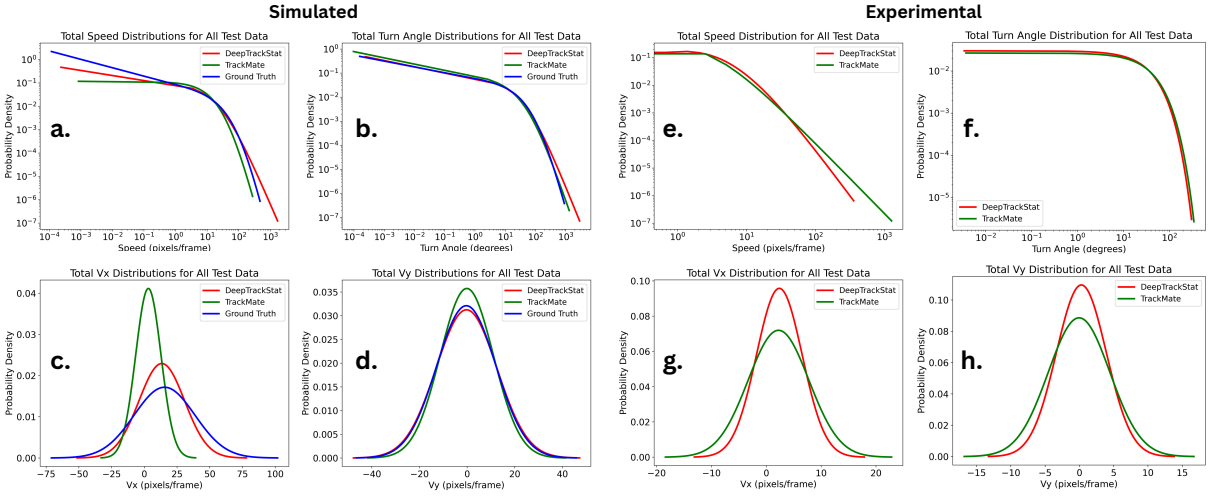

Figure 4: Distribution comparisons for simulated (a-d) and experimental (e-h) test sets for speed (a & e), turn angle (b & f), V_x (c & g), and V_y (d & h). For the simulated test set, we compare DTS and TM to the ground truth. For the experimental test set, we only compare DTS with TM, since there is no ground truth. Each distribution is obtained from a concatenated list of all values (from each individual simulation) for the respective statistic.

Table 8
Average results from the experimental test set (n=23 samples) for each statistic. Since there is no ground truth data for the experimental data, errors for DTS are calculated relative to TM.

Stat	MAE	RMSE	W1
S	1.98 [.312, 4.19]	5.36 [2.17, 11.8]	.026 [.006, .053]
V_x	1.26 [.072, 2.11]	2.29 [.757, 4.38]	.026 [.0056, .058]
V_y	.298 [.008, .793]	1.89 [.407, 2.95]	.033 [.002, .073]
α	7.05 [.724, 14.1]	10.8 [4.12, 17.1]	.001 [.00006, .002]

3.4. Experimental test set

The results from the experimental test set (Table 8) further indicate DTS' ability to accurately extract motion statistics from videos of moving particles. DTS shows strong alignment with TM for all statistics. The values are close enough that, given DTS' strong performance on simulated test set, it's unclear which distributions are more accurate. For example, the high speeds predicted by TM (Fig. 4e), such as up to 1000 pixels/frame, are highly unlikely for a video of bacteria in porous media flows at 2048x2048 resolution. This would mean that a particle could move across the camera's field of view in only two frames, which is not possible for the max flow speed (up to $800 \mu m/s$), frame rate (10 FPS), image magnification (0.325 px/μm) of our specific experiments. A quick calculation shows the max speed a particle should be able to achieve in pixels/frame is about 246, meaning that DTS' estimate of a max speed of about 350 pixels/frame is in all likelihood more accurate than TM's estimate of 1000 pixels/frame.

4. Conclusions

We show that our proposed model, DeepTrackStat, achieves SOTA performance at comparatively rapid speeds for the general task of predicting speed, V_x, V_y, and turn angle distributions for a wide variety of particle tracking situations. Specifically, our model is capable of predicting these motion statistics for a large range of particle, image, and trajectory types (dispersive, Brownian, Poiseuille) about 6 times faster than via classical particle tracking algorithms. Through ablation experiments we show that our novel class-based ensembling method outperforms a simple ensembling method. We then show that DTS outperforms all classical PT algorithms used in this study for the prediction of motion statistics for our simulated test set, and we confirm the applicability of our models to real-world data by showing that the outputs of DTS are comparable to the outputs obtained from TM. In addition, we highlight DTS' strong performance for the specific task of predicting statistics from videos of particles moving at high speeds. In this case, the performance of DTS greatly exceeds that of TM (the top-performing classical algorithm). Thus, we present a novel method for extraction of motion statistics and apply it to videos of particles. Although our models are specifically trained for the task of extracting statistics from videos of moving particles, our class-based ensembling framework can theoretically be extended to extract motion statistics of any set of objects where the speed of the object is significantly correlated with the other motion statistics to be predicted.

Although we have shown the robust performance of DTS across a wide variety of image and motion types, there are many limitations present in our study that primarily revolve around scope. First and foremost, we recognize that a more rigorous study would include more simulated and experimental test sets. In addition, DTS was calibrated on 50% of the simulated test data, so for particles with motion statistics that greatly fall outside of the training or calibration range, it is unlikely that DTS will perform well. This can be seen in the case of the straight trajectories (Fig. 3) - although DTS was trained on trajectories of similar types, straight-trajectory simulations made up a small percentage the entire training set. Thus, without the addition of a manually set flag to indicate that the particles are moving straight, DTS is not able to make accurate turn angle predictions. Another primary limitation of our work is that we have only compared DTS to classical PT algorithms. A more robust study might use fine tuning of SOTA object tracking algorithms to more effectively combine the domains of particle tracking and object tracking and determine more optimal network architectures. Finally, our work is limited in that it can only be used to predict four motion statistics. Rigorous transport studies often need more statistical evidence to make insightful claims, so our work could be improved by increasing the number of statistics DTS can accurately predict.

We hope that our model is of practical use to researchers interested in applications of particle tracking. We have supplied the model weights at https://zenodo.org/records/11245477, and all data and scripts needed for training and testing can be found at https://github.com/mberghouse/DeepTrackStat. Furthermore, our training data represents one of the most comprehensive sets ground-truth particle tracking data publicly available on the internet, and we believe researchers will find it useful for the development of even more robust applications related to PT. Thus, we hope that our work generally sparks interest within the research community about applications of computer vision for particle tracking and motion statistics predictions. Neither of these are solved problems yet, and improving these tasks can greatly improve research capabilities in the wide variety of fields that make use of them.

CRediT authorship contribution statement

Marc Berghouse: Data curation, Formal analysis, Investigation, Software, Conceptualization, Visualization, Writing – original draft, Writing – review & editing, Methodology. Rishi Parashar: Supervision, Writing - review & editing, Project administration.

Declaration of Competing Interests

The authors declare no competing interests.

Acknowledgements

This research is based upon work supported by the U.S. Department of Energy (DOE) under award number DE-SC0019437. The second author (R.P) acknowledges career advancement support in the field of artificial intelligence based methods provided by NSF grant 2123481.

Data Availability

Model weights can be found at https://zenodo.org/records/11245477, and all data and scripts needed for training and testing can be found at https://github.com/mberghouse/DeepTrackStat. For any other inquiries related to data or code please contact the corresponding author.

References

[1] Scheidweiler, D., Miele, F., Peter, H., Battin, T. J. & de Anna, P. Trait-specific dispersal of bacteria in heterogeneous porous environments: from pore to porous medium scale. J. The Royal Soc. Interface 17, 20200046, DOI: 10.1098/rsif.2020.0046 (2020).

[2] Dehkharghani, A., Waisbord, N., Dunkel, J. & Guasto, J. Bacterial scattering in microfluidic crystal flows reveals giant active taylor–aris dispersion. Proc. Natl. Acad. Sci. 116(23), 11119–11124 (2019).

[3] Bhattacharjee, T. & Datta, S. Bacterial hopping and trapping in porous media. Nat. communications 10(1), 2075 (2019).

[4] Dentz, M., Creppy, A., Douarche, C., Clément, E. & Auradou, H. Dispersion of motile bacteria in a porous medium. J. Fluid Mech. 946, A33 (2022).

[5] Creppy, A., Clément, E., Douarche, C., D'angelo, M. & Auradou, H. Effect of motility on the transport of bacteria populations through a porous medium. Phys. Rev. Fluids 4(1), 013102 (2019).

[6] Dehkharghani, A., Waisbord, N. & Guasto, J. S. Self-transport of swimming bacteria is impaired by porous microstructure. Commun. Phys. 6, 18, DOI: 10.1038/s42005-023-01136-w (2023).

[7] Rusconi, R., Guasto, J. S. & Stocker, R. Bacterial transport suppressed by fluid shear. Nat. Phys. 10, 212–217, DOI: 11 10.1038/nphys2883 (2014).

[8] Bhattacharjee, T. & Datta, S. S. Bacterial hopping and trapping in porous media. Nat. Commun. 10, 2075, DOI: 413 10.1038/s41467-019-10115-1 (2019).

[9] Yang, X., Parashar, R., Sund, N.L., Plymale, A.E., Scheibe, T.D., Hu, D. & Kelly, R.T. On modeling ensemble transport of metal reducing motile bacteria. Scientific reports, 9(1), 14638 (2019).

[10] Saxton, M. Single-particle tracking: the distribution of diffusion coefficients. Biophys. journal 72(4), 1744–1753 (1997).

[11] Hong, Q., Sheetz, M. & Elson, E. Single particle tracking. analysis of diffusion and flow in two-dimensional systems. Biophys. journal 60(4), 910–921 (1991).

[12] Forier, K., Messiaen, A.S., Raemdonck, K., Deschout, H., Rejman, J., De Baets, F., Nelis, H., De Smedt, S.C., Demeester, J., Coenye, T. & Braeckmans, K. Transport of nanoparticles in cystic fibrosis sputum and bacterial biofilms by single-particle tracking microscopy. Nanomedicine 8(6), 935–949 (2013).

[13] Jeon, H., Lee, Y., Jin, S., Koo, S., Lee, C.S. & Yoo, J.Y. Quantitative analysis of single bacterial chemotaxis using a linear concentration gradient microchannel. Biomed. microdevices 11, 1135–1143 (2009).

[14] Partridge, J.D., Nhu, N.T., Dufour, Y.S. & Harshey, R.M. Escherichia coli remodels the chemotaxis pathway for swarming. MBio, 10(2), 10-1128 (2019).

[15] Kim, D., Liu, A., Diller, E. & Sitti, M. Chemotactic steering of bacteria propelled microbeads. Biomedical microdevices, 14, 1009-1017 (2012).

[16] Vallotton, P., Van Oijen, A.M., Whitchurch, C.B., Gelfand, V., Yeo, L., Tsiavaliaris, G., Heinrich, S., Dultz, E., Weis, K. & Grünwald, D., 2017. Diatrack particle tracking software: Review of applications and performance evaluation. Traffic 18(12), 840–852 (2017).

[17] Linkhorst, J., Beckmann, T., Go, D., Kuehne, A. J. & Wessling, M. Microfluidic colloid filtration. Sci. reports 6(1), 22376422 (2016).

[18] Dentz, M., Creppy, A., Douarche, C., Clément, E. & Auradou, H. Dispersion of motile bacteria in a porous medium. Journal of Fluid Mechanics, 946, A33 (2022).

[19] Molaei, M. & Sheng, J. Succeed escape: Flow shear promotes tumbling of Escherichia colinear a solid surface. Sci Rep 6, 35290 (2016).

[20] Berghouse, M., Miele, F., Perez, L., Bordoloi, A., Morales, V., & Parashar, R. Evaluation of Particle Tracking Codes for Dispersing Particles in Porous Media. Sci Rep 14, 24094 (2024). https://doi.org/10.1038/s41598-024-75581-0

[21] de Anna, P., Pahlavan, A.A., Yawata, Y., Stocker, R. & Juanes, R. Chemotaxis under flow disorder shapes microbial dispersion in porous media. Nat. Phys. 17, 68–73 (2021).

[22] Kaya, T. & Koser, H. Direct upstream motility in Escherichia coli. Biophysical journal, 102(7), 1514-1523 (2012).

[23] Dehkharghani, A., Waisbord, N. & Guasto, J.S. Self-transport of swimming bacteria is impaired by porous microstructure. Nat. Commun Phys 6, 18 (2023).

[24] Poonoosamy, J., Lu, R., Lönartz, M.I., Deissmann, G., Bosbach, D. and Yang, Y., 2022. A lab on a chip experiment for upscaling diffusivity of evolving porous media. Energies, 15(6), p.2160.

[25] Jahanbakhsh, A., Wlodarczyk, K.L., Hand, D.P., Maier, R.R. and Maroto-Valer, M.M., 2020. Review of microfluidic devices and imaging techniques for fluid flow study in porous geomaterials. Sensors, 20(14), p.4030.

[26] Chenouard, N., Smal, I., De Chaumont, F., Maška, M., Sbalzarini, I.F., Gong, Y., Cardinale, J., Carthel, C., Coraluppi, S., Winter, M. & Cohen, A.R. Objective comparison of particle tracking methods. Nature methods 11(3), 281–289 (2014).

[27] Midtvedt, B., Pineda, J., Skärberg, F., Olsén, E., Bachimanchi, H., Wesén, E., Esbjörner, E.K., Selander, E., Höök, F., Midtvedt, D. & Volpe, G. Single-shot self-supervised particle tracking. arXiv preprint arXiv:2202.13546 (2022).

[28] Midtvedt, B., Pineda, J., Skärberg, F., Olsén, E., Bachimanchi, H., Wesén, E., Esbjörner, E.K., Selander, E., Höök, F., Midtvedt, D. & Volpe, G. Single-shot self-supervised object detection in microscopy. Nature communications, 13(1), 7492 (2022).

[29] Ershov, D., Phan, M.S., Pylvänäinen, J.W., Rigaud, S.U., Le Blanc, L., Charles-Orszag, A., Conway, J.R., Laine, R.F., Roy, N.H., Bonazzi, D. & Duménil, G. Bringing TrackMate into the era of machine-learning and deep-learning. BioRxiv, (2021).

[30] Kapoor, V. & Carabaña, C. Cell Tracking in 3D using deep learning segmentations. In Python in Science Conference, 154-161 (2021).

[31] Schmidt, U., Weigert, M., Broaddus, C. & Myers, G. Cell detection with star-convex polygons. Springer International Publishing. In Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11, 265-273, (2018).

[32] Tinevez, J.Y., Perry, N., Schindelin, J., Hoopes, G.M., Reynolds, G.D., Laplantine, E., Bednarek, S.Y., Shorte, S.L. and Eliceiri, K.W., 2017. TrackMate: An open and extensible platform for single-particle tracking. Methods, 115, pp.80-90.

[33] Fatemi, B., Halcrow, J. & Jaqaman, K. Geometric deep learning of particle motion by MAGIK. Nature Machine Intelligence, 5(5), 483-484 (2023).

[34] Zou, Z., Chen, K., Shi, Z., Guo, Y. and Ye, J., 2023. Object detection in 20 years: A survey. Proceedings of the IEEE, 111(3), pp.257-276.

[35] Amosa, T.I., Sebastian, P., Izhar, L.I., Ibrahim, O., Ayinla, L.S., Bahashwan, A.A., Bala, A. and Samaila, Y.A., 2023. Multi-camera multi-object tracking: a review of current trends and future advances. Neurocomputing, 552, p.126558.

[36] Weller, H., Tabor, G., Jasak, H. & Fureby, C. A tensorial approach to computational continuum mechanics using449 object-oriented techniques. Comput. Phys. 12, 620 (1998).

[37] Zeming, K.K., Salafi, T., Chen, C.H. & Zhang, Y. Asymmetrical deterministic lateral displacement gaps for dual functions of enhanced separation and throughput of red blood cells. Scientific reports, 6(1), 22934 (2016).

[38] Lee, J.H., Fredrickson, J.K., Plymale, A.E., Dohnalkova, A.C., Resch, C.T., McKinley, J.P. & Shi, L. An autotrophic H 2 -oxidizing, nitrate-respiring, Tc(VII)-reducing Acidovorax sp. isolated from a subsurface oxic-anoxic transition zone: H 2 -oxidizing, Tc-reducing Acidovorax spp. Environmental Microbiology Reports 7, 395–403 (2015).

[39] Caccavo Jr, F., Lonergan, D.J., Lovley, D.R., Davis, M., Stolz, J.F. & McInerney, M.J. Geobacter sulfurreducens sp. nov., a hydrogen- and acetate-oxidizing dissimilatory metal-reducing microorganism. Appl Environ Microbiol 60, 3752–3759 (1994).

[40] Ahmed, B., Cao, B., McLean, J.S., Ica, T., Dohnalkova, A., Istanbullu, O., Paksoy, A., Fredrickson, J.K. & Beyenal, H. Fe(III) Reduction and U(VI) Immobilization by Paenibacillus sp. Strain 300A, Isolated from Hanford 300A Subsurface Sediments. Appl Environ Microbiol 78, 8001–8009 (2012).

[41] Hau, H.H. & Gralnick, J.A. Ecology and biotechnology of the genus Shewanella. Annu. Rev. Microbiol., 61, 237-258 (2007).

[42] Ohi, A.Q., Mridha, M.F., Hamid, M.A., Monowar, M.M. and Kateb, F.A. Fabricnet: A fiber recognition architecture using ensemble convnets. IEEE Access, 9, 13224-13236 (2021).

[43] Al-Khateeb, T., Masud, M.M., Khan, L., Aggarwal, C., Han, J. and Thuraisingham, B. Stream classification with recurring and novel class detection using class-based ensemble. In 2012 IEEE 12th international conference on data mining, 31-40, IEEE (2012).

[44] Ali, M., Zhu, P., Jiang, R., Huolin, M., Ehsan, M., Hussain, W., Zhang, H., Ashraf, U. and Ullaah, J. Reservoir characterization through comprehensive modeling of elastic logs prediction in heterogeneous rocks using unsupervised clustering and class-based ensemble machine learning. Applied Soft Computing, 148, 110843 (2023).

[45] Yuan, L., Hou, Q., Jiang, Z., Feng, J. & Yan, S. Volo: Vision outlooker for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 45(5), 6575-6586 (2022).

[46] Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P. & Shao, L. Pvt v2: Improved baselines with pyramid vision transformer. Computational Visual Media, 8(3), 415-424 (2022).

[47] Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., & Dollár, P.: Designing Network Design Spaces. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10425-10433, Seattle, WA, USA (2020).

[48] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R. & Fei-Fei, L. Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 1725-1732 (2014).

[49] Chen, J. & Ho, C.M. MM-ViT: Multi-modal video transformer for compressed video action recognition. In Proceedings of the IEEE/CVF winter conference on applications of computer vision, 1910-1921 (2022).

[50] Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M. & Schmid, C. Vivit: A video vision transformer. In Proceedings of the IEEE/CVF international conference on computer vision, 6836-6846 (2021).

[51] Li, Y., Wu, C.Y., Fan, H., Mangalam, K., Xiong, B., Malik, J. & Feichtenhofer, C. Mvitv2: Improved multiscale vision transformers for classification and detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4804-4814 (2022).

[52] Ross Wightman. PyTorch Image Models. Github, Github repository (2019). doi: 10.5281/zenodo.4414861.

[53] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, 248-255 (2009).

[54] Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2623-2631 (2019).

[55] Allan, D. B. Trackpy. "soft-matter/trackpy: v0.6.4". Zenodo (2024). doi: 10.5281/zenodo.12708864.

[56] Heyman, J. TracTrac: A fast multi-object tracking algorithm for motion estimation. Computers & Geosciences, 128, 11-18 (2019).

[57] Fukai, Y.T. and Kawaguchi, K. LapTrack: linear assignment particle tracking with tunable metrics. Bioinformatics, 39(1), 799 (2023).

[58] Abràmoff, M.D., Magalhães, P.J. and Ram, S.J., 2004. Image processing with ImageJ. Biophotonics international, 11(7), 36-42, (2004).

[59] Harley, A.W., Fang, Z., Fragkiadaki, K. Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. In ECCV (2022).