Spline-based Decomposition of Streamed Particle Trajectories for Efficient Transfer and Analysis

K. Scharnowski[1] and S. Frey[1] and B. Raffin[2] and T. Ertl[1]

[1]VISUS, University of Stuttgart, Germany
[2]Inria, Grenoble, France

Abstract

We introduce an approach for distributed processing and efficient storage of noisy particle trajectories, and present visual analysis techniques that directly operate on the generated representation. For efficient storage, we decompose individual trajectories into a smooth representation and a high frequency part. Our smooth representation is generated by fitting Hermite Splines to a series of time windows, adhering to a certain error bound. This directly supports scenarios involving in situ and streaming data processing. We show how the individually fitted splines can afterwards be combined into one spline posessing the same mathematical properties, i.e. C^1 continuity as well as our error bound. The fitted splines are typically significantly smaller than the original data, and can therefore be used, e.g., for an online monitoring and analysis of distributed particle simulations. The high frequency part can be used to reconstruct the original data, or could also be discarded in scenarios with limited storage capabilities. Finally, we demonstrate the utility of our smooth representation for different analysis queries using real world data.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction and Motivation

Modern computer simulations, e.g. from Molecular Dynamics Simulation, crucially support the study of different phenomena of interest. However, the increasing availability of computational resources and the resulting amount of data makes related analysis tasks more and more challenging. It can be infeasible to store (or include in the analysis) all generated data due to constrained available storage capacity or memory bandwidth. The most widely employed approach in practice is to sample the simulation results sparsely, potentially leading to an inaccurate analysis. To address this, more sophisticated, parallelizable, methods are needed to reduce the size of the stored data. Those approaches normally either reduce entropy of the data for better compression or extract features of interest [TRB*08].

While numerous methods have been presented to efficiently store general data (c.f. [LI, FM, CCMW]), there are not many approaches specifically targeting particle trajectories while at the same time providing direct support for certain analysis tasks. Both for analysis and efficient storage it is beneficial to decompose the trajectories into a low frequency and a high frequency part. On the one hand, looking at the smooth low frequencies helps with similarity measurements used in clustering algorithms. On the other hand, separating high frequencies is beneficial for reducing data entropy, since they can be quantized in frequency space (similar to video and image compression codecs, such as JPEG. However, using a spectral analysis to separate the low and high frequencies has some disadvantages specifically for the analysis part. Ideally, one wants to control the smoothness of the low frequency part using a error bound, particularly, since it depends highly on the application scenario and the analysis task at hand. For example, Brownian Motion in some cases might be considered as noise, in others as important part of the analysis. This kind of control cannot be provided when using pure spectral analysis. Due to the nature of Discrete Fourier Analysis, aliasing is likely to appear at the borders of the low frequency part between the streamed time windows, if they are to be treated independently. Consequently, the consecutive time windows need to be merged afterwards while still providing the same error bound. In order to extract the low frequency part from the individual trajectories, curve fitting models can be used [DC, LSE*].

In this paper, we present an approach for the decomposition of trajectory data with splines, including user-controlled smoothness, and frequency based representations. For representing the low frequency part, we use C^1-continuous Hermite Splines. Our approach can be applied to individual time windows indipendently, allowing for parallelization of the computations, while at the same time allowing for a later combination of the splines associated with the time windows to one C^1-continuous spline. Crucially, we obtain analytical definitions of the first derivative, which facilitates certain analysis tasks, such as clustering algorithms. Depending on the desired outcome, the original data can be fully reconstructed by

Algorithm 1 *Distributed spline-based trajectory decomposition approach running independently on each node.*

```
▷ this runs on each node (i.e., for each space partition)
procedure DISTRIBUTEDTRAJECTORYGENERATION
    for all time windows T do
        Ψ^T ←fit spline (x_i^T, σ)
        if reconstruct full spline then
            G^T ← store boundary positions
        end if
        if do point reconstruction then
            F^T ← DFT (x_i^T − Ψ^T)
        end if
    end for
end procedure
```

Algorithm 2 *Trajectory reconstruction and analysis. This runs on for the data collected from each node (generated via Alg. 1).*

```
▷ this runs on collected data from each node (cf. Alg. 1)
procedure TRAJECTORYMERGINGANALYSIS
    for all p ∈ P do
        if reconstruct full spline then
            Ψ ← combine splines (Ψ^T, G^T; ∀T)
        end if
        if do point reconstruction then
            reconstruct positions (Ψ^T, F^T; ∀T)
        end if
    end for
    do analysis
end procedure
```

compressing and storing the high frequency part and later adding it to the low frequency part. In particular, we contribute the following:

- An approach designed to support streaming and in situ processing (Sec. 2.1)
- A smooth and compact representation of particle trajectories by fitting of Hermite Splines with user-controlled error (Sec. 2.2)
- The generation of one consistent spline (adhering to the same error criterion) a posteriori across different time windows (Sec. 2.3)
- The spline is useful for certain analysis tasks due to its generally high quality and favorable properties (Sec. 3)

2. Methodology

Our algorithm was designed to support processing of streamed time windows while at the same time efficiently supporting inherently global (visual) analysis operations.

2.1. Approach Structure For In Situ Processing

Our approach is conceptually structured into two phases: **(a)** an in situ/streaming part decomposes trajectory time windows into our spline-based representation, and **(b)** a post-hoc part that assembles individual time windows into one consistent representation.

(a) In situ Trajectory Decomposition (Alg. 1). We encode subsets of the particles and for those subsets again separate time windows completely independently of each other. Time windows might originate in a streaming setup, in which one decides to start the encoding process once a certain number of positions has been received for a trajectory. For each time window, we fit a spline through the respective positions using a user-defined mean error bound σ. For numerous scenarios, this spline suffices already, but in case a full reconstruction of the original data is required, we create a frequency representation of the difference between the spline and the original points. In theory, this process allows to work fully independently and in parallel on each particle trajectory for each time window.

(b) Trajectory Reconstruction and Analysis (Alg. 2). Our spline-based representation generated in **(a)** can then be used for the visual analysis of the data. Generally, a user has two basic options (which may also be combined). First, if high frequencies have been stored, the original point data can be reconstructed in combination with the time window splines. Second, for each particle p_i, a combined spline can be generated from the various time window splines. This spline representing the whole trajectory is both smooth and $\mathcal{C}_1$ continuous, which is not only favorable for its direct visualization, but it is also directly usable for a variety of different analysis tasks.

2.2. Treatment of individual Time-windows

Given a particle p_i, we call its time-dependent position $\mathbf{x}_i^T$ its *trajectory*, with T being the time window, in which the trajectory is defined. Throughout this document we will refer to a set of particles as $\mathcal{P}$ and to the respective set of trajectories as $\mathcal{X}^T$. In order to decompose a trajectory $\mathbf{x}_i^T$ into a low frequency part and a high frequency part with direct error control, we fit a hermite spline to the series of points, representing the trajectory of one particle. Our goal is to obtain a spline for each time window, so that we can merge them in a post-processing step ((Fig. 1a).

The fitting is done by minimizing the error between the current spline and the respective trajectory. We compute this error by uniformly sampling the spline at n locations, n being the number of time steps in the treated time window. We then compute the *Root Mean Square Deviation (RMSD)* between the two point series:

$$RMSD = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\delta_i^2}, \tag{1}$$

where δ is the distance between two associated points.

For error control, a mean error bound σ is provided by the user, who can choose it depending on the application scenario and analysis task. We start by a Hermite Spline that consists of a single linear segment, that we initialize with the first and last points of the trajectory. Subsequently, we fit this spline to the trajectory by minimizing our cost function. The variables in this fitting process are the position and slope at each control point of the current spline. If, after the fitting, the cost function is below σ, we stop here. Otherwise, we subdivide the spline into more segments and fit again. Subdivision without changing the shape of the current Hermite Spline can be achieved by cutting all segments in half and scaling the slope of all control vertices with 0.5. Doing so allows us to maintain the uniform, implicit parameterization of the spline, which we would have

to store separately otherwise. This process, consisting of subsequent subdivision and fitting is repeated until the error criterion is met.

The resulting spline delivers an analytical representation of the low frequencies of the trajectory. The high frequencies are subsequently calculated by subtraction from the associated spline locations.

2.3. Merging Consecutive Time Windows

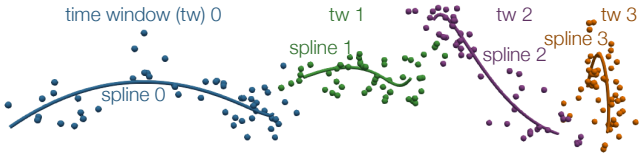
When using our approach for treating individual time windows as described in Section 2.2, we can afterwards merge the individual splines to one combined spline possessing the same favorable properties, namely, $\mathcal{C}^1$ continuity and a guaranteed mean error below σ. We achieve this by connecting the splines of consecutive time windows with transition splines (Fig. 1b & c). We could just connect the endpoints of the respective time window splines with one segment, defined by the endpoints slope and position. While this would give us the desired $\mathcal{C}^1$-contiuity, it could potentially lead to a spline that largely overfits the original data. We, therefore, need to preserve a small part of the original trajectory positions when encoding, so we can use them later for the transition spline fitting. To this end, we explicitly store m positions at the beginning and the end of each time window. Since we have time windows with hundreds of time steps, this is only a small portion. Particularly, in our experiments setting $m = 5$ was sufficient. The portions of the time window splines associated with these transition points are cut off and discarded for the full spline reconstruction. The transition spline is then initialized with the new end position and slope of the time windows and subsequently fitted to the transition points using subdivision. The dfference to the previous fitting process is here, that the position and slope of the endpoints are kept fixed.

3. Results

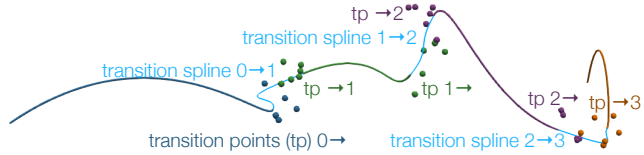
We developed two separate tools for both the encoding and the post-processing stage. The prototypes were developed in Python and all pictures in this Section were created using ParaView [Aya15]. We tested our approach using two data sets from different application domains. All computations were done on a desktop mache with an Intel Core i7-4770 (3.4GHz) and 16 GB RAM. The encoding of the data into our format took roughly 15 minutes for the first data set and 3 minutes for the second data set.

The first data set is the result of a simulation of a protein (Lysozyme) in water solvent. The simulation was executed based on the publicly available tutorial using Gromacs [Lem22] and was sampled every 1ps leading to 200 frames in total. The reconstructed splines can be found in Fig. 2. Please note, that we basically removed the periodic boundary conditions of the data in order to facilitate the fitting. We see a notable difference between the protein atoms and the atoms belonging to solvent molecules. As expected, the solvent molecules move in a much faster and more chaotic way. The trajectory traces of individual water molecules appear as bundles of three traces each. This illustrates that our method is capable of creat similarspline representations for similar input data.

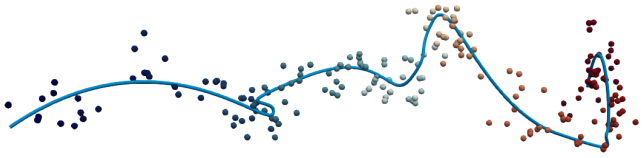
The second data set is a measured Fiber experiment created via Digital Image Correlation (DIC) where properties of elastic material are investigated. Measurement locations are positioned on the



(a) *independent construction of splines in different time windows*



(b) *merging of consecutive time windows*



(c) *full $\mathcal{C}_1$-continuous spline fitting to the complete trajectory*

Figure 1: *Different stages of a single particle trajectory crossing four consecutive time windows. (a) illustrates the division of the trajectory into four time windows and depicts the splines that are fitted for each time window. (b) shows the stored transition points that are then used to fit transition splines in between the time window specific splines. (c) shows the full $\mathcal{C}_1$-continuous spline fitting to the complete trajectory while maintaining the error bound as well as wanted mathematical properties.*

material patch and tracked while the material is stretched, leading to a series of positions for each locations. In this case, the points are defined in 2D and the trajectory is 254 frames long. This data possesses some fundamentally different properties in comparison to the simulation data. Specifically, we wanted to investigate the potential of our spline representation for analysis tasks related to clustering of particle trajectories. Clustering requires a suitable similarity function to cluster the traces. To this end, we used an analysis step that is inspired by the bag-of-features approach used in computer visison [LWS13]. First, we sample the gradient magnitude of each trajectory path and compute the histogram. We then use the *Earth Mover's Distance* as a similarity measurement. The resulting clusters can be seen in Figure 3. The spatial distribution of the clusters depicts that the there is similarities in neighbouring traces, indicating that they have a similar distribution of speed.

4. Discussion & Conclusion

We described an approach for decomposing particle trajectories using Hermite Splines. Our method can easily be applied to partitioned data in a streaming fashion, making it suitable for in situ and out-of-core application scenarios. We furthermore showed, how the resulting data representation can be used to facilitate analysis and potentially gain new insights.

The most important parameter for our approach is the spline mean error σ, which is used to fit a spline to each time windows and is also considered during merging. This parameter allows domain domains

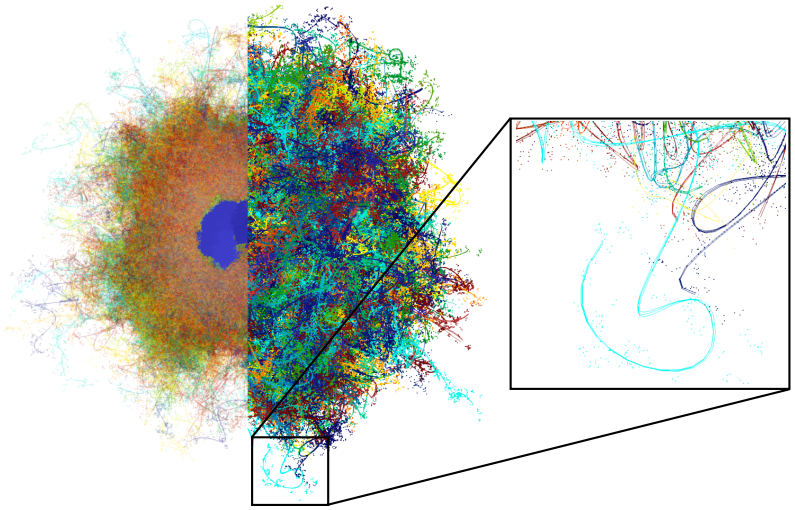


Figure 2: *Reconstructed splines for protein in water data. The left part depicts the different behaviour of the protein atoms (opaque) and the solvent molecules (transparent). The solvent molecules show strong movement while the protein stays relatively rigid. The cutout (right) shows that the bundled traces of the three atoms belonging to one water molecule.*

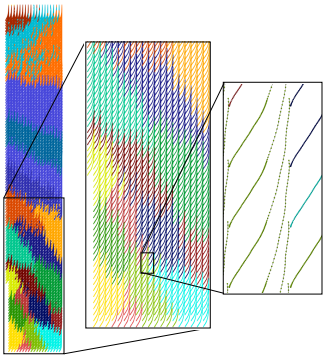


Figure 3: *Analysis of experimental data using clustering.*

scientists to define the smoothness that their data should have for later analysis. However, the parameter also heavily influences the computation time for our algorithm. Naturally, a higher mean error leads to faster convergence and, therefore, less computational effort. This is due to the fact, that the spline than needs less segments to meat the requirements. This computation time can also be lowered by choosing a smaller time window size. However, a too small time window size leads to less efficient storage since the spline representation is then prone to creating a storage overhead.

This paper presents ongoing research. For future work, since the high frequencies are likely to have rather small entropy, we aim to compress them using respective encoding and quantization. Additionally, we want to investigate how beneficial the properties of our decomposition are in this respect, hopefully leading to good compression schemes. Furthermore, we want to test our approach in a distributed environment. To this end, we can easily extend our time window definition to 4D space time blocks. This would allow us to use our approach in an insitu context and, furthermore, investigate the scalability of the algorithm to big data.

Acknowledgements

We want to thank Ilyass Tabiai and Patrick Diehl from the Laboratory for Multiscale Mechanics of the Polytechnique Montréal for providing the Fiber experiment data and background information. We further thank the German Research Foundation (DFG) for supporting the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

References

[Aya15] AYACHIT U.: *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., USA, 2015. 3

[CCMW] CHEN D., CHIANG Y.-J., MEMON N. D., WU X.: Lossless geometry compression for steady-state and time-varying irregular grids. In *EuroVis*. 1

[DC] DI S., CAPPELLO F.: Fast error-bounded lossy hpc data compression with sz. In *IPDPS 2016*, IEEE, IEEE. 1

[FM] FOUT N., MA K.-L.: An adaptive prediction-based approach to lossless compression of floating-point volume data. 2295–2304. 1

[Lem22] LEMKUL J.: GROMACS Tutorial - Lysozyme in Water. http://www.bevanlab.biochem.vt.edu/Pages/Personal/justin/gmx-tutorials/lysozyme_old/index.html, 2014 (accessed 2016-12-22). 3

[LI] LINDSTROM P., ISENBURG M.: Fast and efficient compression of floating-point data. 1245–1250. 1

[LSE*] LAKSHMINARASIMHAN S., SHAH N., ETHIER S., KLASKY S., LATHAM R., ROSS R., SAMATOVA N. F.: *Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-temporal Data*. Springer Berlin Heidelberg, pp. 366–379. 1

[LWS13] LI Y., WANG C., SHENE C.-K.: Streamline similarity analysis using bag-of-features, 2013. 3

[TRB*08] TU T., RENDLEMAN C. A., BORHANI D. W., DROR R. O., GULLINGSRUD J., JENSEN M. O., KLEPEIS J. L., MARAGAKIS P., MILLER P., STAFFORD K. A., SHAW D. E.: A scalable parallel framework for analyzing terascale molecular dynamics simulation trajectories. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing* (Piscataway, NJ, USA, 2008), SC '08, IEEE Press, pp. 56:1–56:12. 1