Complexity Estimation for Feature Tracking Data

Dirk N. Helmrich*, Andrea Schnorr*, Torsten W. Kuhlen*, and Bernd Hentschel*

JARA – High-Performance Computing, Aachen, Germany
Visual Computing Institute, RWTH Aachen University, Germany

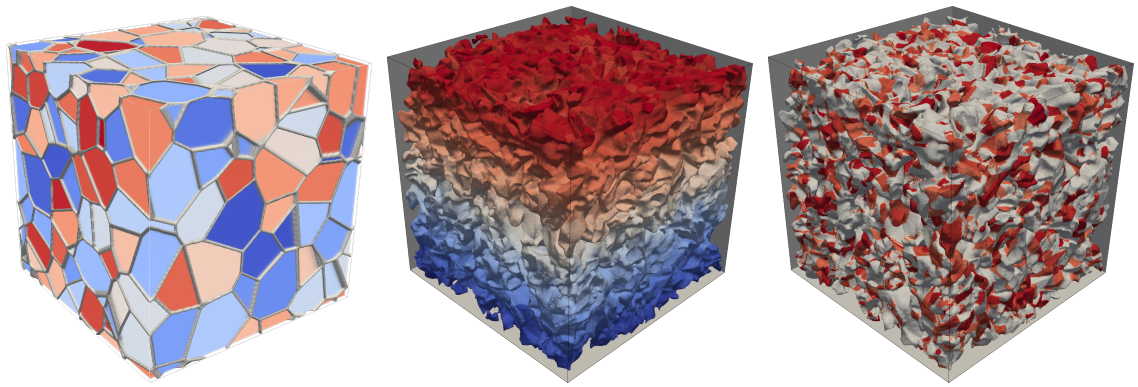

Figure 1: Left: Synthetic Voronoi data set to mimick the behavior of dissipation elements. Points are seeded randomly inside this data set and each volume cell is labeled with its cell center. These are colored by cell center ID. Middle: Dissipation element data set, colored by ID Right: Dissipation element data set, colored by predicted complexity of the problem

Abstract

Feature tracking is a method of time-varying data analysis. Due to the complexity of the underlying problem, different feature tracking algorithms have different levels of correctness in certain use cases. However, there is no efficient way to evaluate their performance on simulation data since there is no ground-truth easily obtainable. Synthetic data is a way to ensure a minimum level of correctness, though there are limits to their expressiveness when comparing the results to simulation data. To close this gap, we calculate a synthetic data set and use its results to extract a hypothesis about the algorithm performance that we can apply to simulation data.

Index Terms: Human-centered computing—Visualization—Visualization design and evaluation methods

1 Introduction

Feature tracking algorithms are means to connect time-varying regions over a series of discrete time steps. These structures are explicitly extracted in each time step, and tracking algorithms are used to determine their temporal evolution. That way, time-depended phenomena can be visualized. The main terminology of feature tracking was developed by Samtaney et al. [4] and Silver et al. [6, 7] and was since used in most publications. Recent work has focused on speeding up the process of feature tracking, or making the results more reliable and provide visualizations for them. Sauer et al. [5] used particle trajectories to enable a fast approximation of subsequent feature objects even if time steps are skipped. Oster et al. [2] created a tracking approach for flame surfaces which utilizes implicit surface tracking to enable the in-situ based assignment computation, i.e., doing a feature tracking analysis step during the runtime of the simulation. They evaluated their results using an analytic function for which the results are deterministic.

Here, we focus on the evaluation of feature tracking algorithms, or more specific; we predict algorithm behavior and correctness on synthetic data and apply the model to simulation data. To do this, we create a 3D-Voronoi graph in which we consider each Voronoi cell as feature object. Through point manipulation, we can produce deterministic temporal behavior, such as movement, and even splits and merges. For the Voronoi graph, there is a ground truth against which to compare tracking algorithms. We describe the feature tracking problem for each feature object statistically, making it possible to apply its predicted complexity to a cubic isotropic turbulence data set. This simulation data set consists of 256^3 cubes of space-filling dissipation elements [3], which are numerically defined as regions whose point trajectories end and begin in the same min/max pair within the data set. It is shown in Fig. 1, middle.

2 Data Generation

To generate an acceptable data basis, an evaluation data set was created that consists of a 256^3-cube filled with Voronoi cells, as shown in Fig. 1, left. These Voronoi cells are generated from a uniformly distributed point set. Feature movement and events are created by point manipulation and point displacement. To mimic a feature split, we seed a point very close to another one, which splits the Voronoi cell in half. For a merge event, two points move close to another and when their distance has reached a threshold, the points are collapsed into one, making the Voronoi cells merge together. Note that Voronoi graphs, just like dissipation elements are inherently space-filling. This means that mimicking death or birth events may not be needed. For a death event, the space of a disappearing element is immediately taken up by neighboring features and as such, a merge is the more viable explanation.

To be able to viably ensure that our complexity estimation is useful, we use a widely-accepted and published tracking algorithm by Silver et al. [6]. The paper uses a threshold-bounded local optimiza-

*E-mail: {helmrich, schnorr, kuhlen, hentschel}@vr.rwth-aachen.de

2018 IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)
21 October 2018, Berlin, Germany
978-1-5386-6873-3/18/$31.00 ©2018 IEEE

tion based on its difference measure. We can determine algorithm behavior, based on an ordering of elements, and taking into account the threshold-bounded local optimization they use, by determining whether the ground truth is the local optimum. Without loss of generality, we fix an ordering $\mathbb{P} = \{i_0, i_1, \cdots, i_n\}$ on the feature object set and a threshold τ. The ordering is used to determine whether the local optimum can be reached or if its corresponding feature objects were already removed from the search space. The tracking algorithm by Silver et al. uses the normalized maximum set difference as a dissimilarity measure between a feature object and an overlapping combination of features. The configuration depends largely on a threshold value $\tau_{\max}$, since the local minimum is only chosen if smaller than $\tau_{\max}$. Obviously, the predicted complexity of the simulation data will also depend on the threshold value chosen. For most feature tracking algorithms, even if there is a prediction step involved, we can calculate some kind of feature object similarity that describes the tracking problem in an expedient fashion.

Since we know in what way the algorithm behaves, it is easier to ensure that the complexity mapping is correct. We run the tracking algorithm on the synthetic 3D-Voronoi graph and quantify its error rate. It is calculated by the ratio of the detected correspondences to all within the ground truth, times the ratio of valid correspondences to all found by the algorithm. For each feature object and its possible successors, we approximate a kernel distribution K, for which we calculate mean, variance, minimum and maximum. Regarding only the local optimization itself, these values provide a good problem description. This can be extended to include error sources like numerical instability by, e.g., providing a normalized size or shape indicator. We also calculate these values for dissipation elements and each combination of feature objects within the simulation data.

3 Error Rate Analysis

Using this calculation as a learning basis, and taking into account the weighting and order of each feature object, we investigated multiple approximation models for their validity and accuracy. For the final mapping, we use a boosted decision tree model [1] that had a root-mean-square deviation of 0.074 on a data set of 240,000 Voronoi cells over 1,000 time steps. The runtimes of the process are depicted in Tab. 1. This regression model contains a hypothesis about which factors contribute to the complexity of the correspondence problem. Using the trained model, we apply its predictions to the dissipation element data. In Fig. 1, right, such a mapping is shown, where a grey region is predicted to be quite simple to track, while a red region is more difficult. The overall data set is rated as quite simple, though there are spots of difficult regions throughout the data set. Note that the algorithm by Samtaney et al. is indeed order depended, however, we did not use this information to train the predictor.

To inspect the validity of our prediction, we inspect individual dissipation elements from the simulation data, especially those that were predicted to be difficult to track. While there is no ground truth for the simulation data, we can inspect certain feature object combinations for which we know that the algorithm will behave in a certain way. In this fashion, we can check if the complexity predictor holds true for this case and if it is a viable technique to pre-scan data sets based on a feature tracking approach for evaluation purposes. By manual inspection, we can determine whether the algorithm's solution is likely to be correct or not. Closer inspection of the marked areas shows that the error rate in complex regions indeed seems to be higher. Fig. 2 shows such an error. The corresponding element is shown in red and its chosen successor is shown in grey. Figure 2 shows that there is, indeed, a large area for which there should be a successor. Since this successor was not found, the predictor correctly marked this dissipation element as hard to track. While a such a turbulence is a quite abstract use-case, it shows that synthetic data sets, even simple ones such as Voronoi graphs, can be used to inspect algorithms for their behavior in real-world data. Furthermore, such a mapping of problem complexity on the simulation space can provide insight into the algorithm and the way the data set itself behaves. Using a more sophisticated algorithm as basis, it would enable inspecting the simulation data for numerical instability or highly complex behavior.

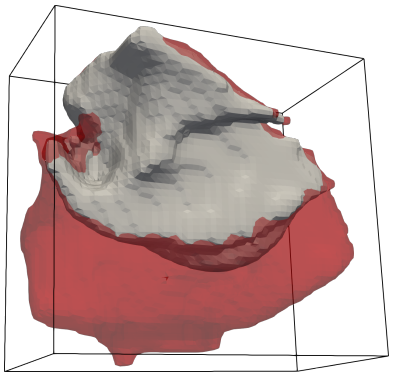

Figure 2: A dissipation element from the simulation data (red) which is predicted to be quite hard to track. The gray region is the only associated feature object. Note that there is a red region for which no successor has been found.

Table 1: Runtimes of the analysis

Volume Data Processing	4468.8 ms
Data Preparation	1766.19 ms
Regression Learner	134.3 s
Model Applying	7781 ms

4 Conclusion

We used a fairly medium-sized synthetic data set to calculate a model that predicts problem complexity from kernel value distributions. We applied this model to a standard data set in which we inspected the validity of the mapping. We showed that this is a viable approach to inspect algorithm behavior on simulation data, which can be extended beyond evaluation.

Acknowledgments

The authors would like to acknowledge the funding provided by the DFG under grant KU 1132/10-1, and the German Excellence Initiative, JARA High-Performance Computing.

References

[1] C. M. Bishop. Pattern recognition and machine learning (information science and statistics).

[2] T. Oster, A. Abdelsamie, M. Motejat, T. Gerrits, C. Rössl, D. Thévenin, and H. Theisel. On-the-fly tracking of flame surfaces for the visual analysis of combustion processes. In *Computer Graphics Forum*, vol. 37, pp. 358–369. Wiley Online Library, 2018.

[3] N. Peters and L. Wang. Dissipation element analysis of scalar fields in turbulence. *Comptes Rendus Mécanique*, 334(8):493–506, 2006.

[4] R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing Features and Tracking Their Evolution. *IEEE Computer*, 27(7):20–27, 1994. doi: 10.1109/2.299407

[5] F. Sauer, H. Yu, and K.-L. Ma. Trajectory-Based Flow Feature Tracking in Joint Particle/Volume Datasets. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2565–2574, Dec. 2014. doi: 10.1109/TVCG.2014.2346423

[6] D. Silver and X. Wang. Volume Tracking. In *Proceedings of IEEE Visualization*, pp. 157 – 164, 1996. doi: 10.1109/VISUAL.1996.567807

[7] D. Silver and X. Wang. Tracking and Visualizing Turbulent 3D Features. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):129–141, 1997. doi: 10.1109/2945.597796