Report of the TopoInVis TTK Hackathon: Experiences, Lessons Learned, and Perspectives

Jonas Lukasczyk, Jakob Beran, Wito Engelke, Martin Falk, Anke Friederici, Christoph Garth, Lutz Hofmann, Ingrid Hotz, Petar Hristov, Wiebke Köpp, Talha Bin Masood, Małgorzata Olejniczak, Paul Rosen, Jan-Tobias Sohns, Tino Weinkauf, Kilian Werner, Julien Tierny

Abstract This paper documents the organization, the execution, and the results of the Topology ToolKit (TTK) hackathon that took place at the TopoInVis 2019 conference. The primary goal of the hackathon was to promote TTK in our research community as a unified software development platform for topology-based data analysis algorithms. To this end, participants were first introduced to the structure and capabilities of TTK, and then worked on their own TTK-related projects while being mentored by senior TTK developers. Notable outcomes of the hackathon were first steps towards Python and Docker packages, further integration of TTK in Inviwo, the extension of TTK with new algorithms, and the discovery of current limitations of TTK as well as future development directions.

Jonas Lukasczyk, Christoph Garth, Jan-Tobias Sohns, Kilian Werner
Technische Universität Kaiserslautern, e-mail: jl@jluk.de, garth@cs.uni-kl.de, j_sohns12@cs.uni-kl.de, kwerner@cs.uni-kl.de

Jakob Beran
Stockholm University, e-mail: jakob.beran@misu.su.se

Wito Engelke, Martin Falk, Ingrid Hotz, Talha Bin Masood
Linköping University, e-mail: wito.engelke@liu.se, martin.falk@liu.se, ingrid.hotz@liu.se, talha.bin.masood@liu.se

Anke Friederici, Wiebke Köpp, Tino Weinkauf
KTH Royal Institute of Technology, e-mail: ankef@kth.se, wiebkek@kth.se, weinkauf@kth.se

Lutz Hofmann
Heidelberg University, e-mail: lutz.hofmann@iwr.uni-heidelberg.de

Petar Hristov
University of Leeds, e-mail: mm16pgh@leeds.ac.uk

Małgorzata Olejniczak
University of Warsaw, e-mail: malgorzata.olejniczak@cent.uw.edu.pl

Paul Rosen
University of South Florida, e-mail: prosen@usf.edu

Julien Tierny
CNRS, Sorbonne Universite, e-mail: julien.tierny@sorbonne-universite.fr

1 Introduction

The Topology ToolKit (TTK) [17] is an open-source software library for topological data analysis (TDA) and scientific visualization. At the time of writing, TTK consists of more than 60 modules—contributed from various researchers from several institutions—that provide efficient algorithms to compute contour trees [7], Reeb graphs [8], persistence diagrams [4, 3], topological simplifications [18], Morse-Smale complexes [15], nested tracking graphs [13, 11], fiber surfaces [10], image-based geometry reconstructions [12], and many more TDA products. The core feature of TTK is its efficient and unified approach to topological data representation that makes it possible to coherently chain these different algorithms. Therefore, developers can contribute new algorithms to TTK in a modular fashion, and end-users can utilize TTK as a production tool for interactive TDA (Fig. 1). Furthermore, this unified approach makes it possible for researchers to reproduce results, benchmark algorithms, and develop new algorithms in an existing interrelated software environment.

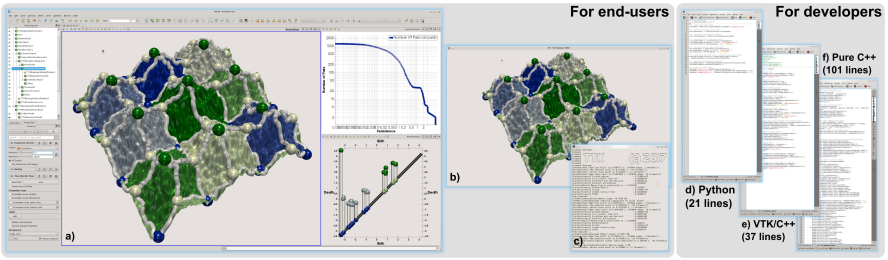

Fig. 1 TTK is a software platform for topological data analysis and scientific visualization. It is both accessible to end-users—via ParaView plugins (a), VTK-based generic GUIs (b), and command-line programs (c)—and to developers—via Python (d), VTK/C++ (e) or dependence-free C++ (f) bindings. TTK provides an efficient and unified approach to topological data representation and simplification, which enables in this example a discrete Morse-Smale complex (a) to comply to the level of simplification dictated by a piecewise linear persistence diagram (bottom-right linked view, a). Code snippets are provided (d-f) to reproduce this pipeline.

However, TTK has two major limitations: a) TTK is difficult to use and extend due to its extensive capabilities and complex software architecture; and b) TTK is not easily accessible since it requires manual compilation and requires several dependencies to utilize all features (which makes it especially difficult to install TTK on non UNIX systems). These limitations often discourage new developers and end-users to utilize TTK in their projects.

In an effort to overcome these limitations and simultaneously grow TTK's user and developer base, two senior TTK developers (Julien Tierny and Jonas Lukasczyk) organized a hackathon as an event in which they would be able to directly interact with people from the topology-based visualization research community, understand their practical needs, and address the concrete problems they face during integration of

TTK in their own projects. This event would therefore be fundamentally different to previous TTK related workshops, such as the TTK tutorials at IEEE VIS in 2018 [6] and 2019 [5] that followed a teacher-centered teaching approach. At the same time, the TopoInVis conference series was experimenting with novel initiatives to increase interest and collaboration within the same community. Thus, the hackathon became part of said initiative, as it seemed—and proved to be—beneficial for both the conference organizers and conference participants to organize the hackathon as a co-located conference event. However, the hackathon organizers did not have first-hand experience about organizing and conducting such an event. Therefore, this work documents

- the organizational aspects of the hackathon that can be used to successfully conduct similar events in the future (Sec. 2); and
- the practical results of the hackathon that actually advanced TTK (Sec. 3).

2 Organization

The hackathon organizers had no prior experience about hosting hackathons; including getting people interested, coping with different backgrounds of the participants, determining a schedule, and setting reasonable goals for such an event. It was only clear that the hackathon will be a single day event that will be co-located with TopoInVis, and that the hackathon should focus on the specific problems developers and end-users encounter when they try to integrate TTK within their own projects.

2.1 Preparation

To get an initial sense on the number, the different experience levels, and the interests of potential participants, the organizers provided a five-minute online survey that consisted of ten multiple choice questions. Thirteen people completed the survey and they all registered for the hackathon. Participants were first asked about their familiarity with TDA and TTK, as well as their programming skills (Fig.2), which indicated that the hackathon program needs to reflect the diverse backgrounds of the participants. Next, each participant was asked to select two of the following suggested hackathon topics she or he is interested in (total number of votes are shown in parenthesis):

- (6) TTK integration into an existing system
- (5) Actual Data Analysis with TTK
- (5) TTK support for vector fields
- (4) TTK support for periodic grids
- (4) TTK packaging
- (2) TTK support for tensor fields

Two participants also used the option to suggest a new topic: the portation of an existing algorithm to TTK. This poll also indicated that participants are interested in largely different topics, so the hackathon program should not focus on a single subject. The remaining questions covered the types of datasets participants commonly have to process, if they plan to bring these datasets to the hackathon, and minor organisational issues.

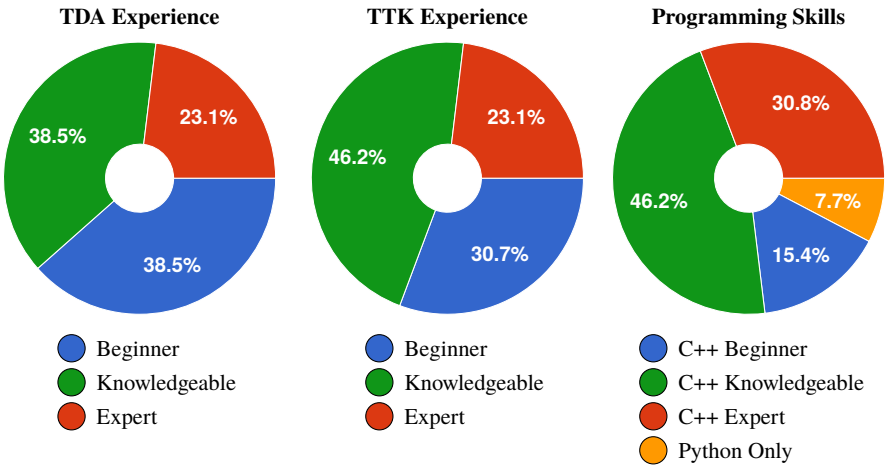

Fig. 2 Experience levels of the 13 participants in regard to topological data analysis (a), TTK (b), and programming languages (c).

2.2 Program

The hackathon was designed as a single-day event that consisted of four sessions (Fig. 3) led by both organizers. To address the diverse backgrounds of the participants (Sec. 2.1), the organizers presented in the first session the internal structure of TTK and demonstrated its capabilities on several examples, where it was possible for participants to follow along if they already had TTK installed.

The introduction provided a common basis for the rest of the hackathon where participants split into small groups consisting of 4-5 people to work on a specific, self-determined coding project based on the topics indicated in the survey. The overall goal of the coding projects was to help participants with initial steps towards first results, and to teach them how to continue these projects themselves later on, as it was already anticipated by the organizers beforehand that most projects can not be completed in one day. Initially, the organizers intended to provide a more extensive introduction that would cover the first two sessions, but due to the experience level and interest of the participants the introduction was cut short to have more time for the actual coding projects.

In the last session, a representative of each group summarized their corresponding achievements of the day, the encountered problems, and the next steps they have to take in order to complete their projects. To this end, every representative had to give a ten minute presentation, followed by a discussion.

08:00 - 10:00	Introduction
10:30 - 12:30	Workgroups
14:00 - 16:00	Workgroups
16:30 - 18:00	Conclusions

Fig. 3 Schedule of the TTK hackathon: After providing an overview of TTK's structure and usage, the participants split into groups that worked on different tasks, and subsequently presented their results in a concluding session.

3 Results

This section presents the most prominent results of the workgroups including improved accessibility, ported algorithms, the integration of TTK in the production tool Inviwo [9], and the support of periodic grids.

3.1 Packaging

An often problematic aspect of scientific data analysis and visualization is the maintenance of software environments, i.e. creating a working installation of an analysis package and all its dependencies. Since TTK is shipped with a rich API (C++, VTK, Python) and a plugin for the production visualization platform ParaView, its full-option compilation can be challenging for novice users. Moreover, this process differs substantially between platforms. The resulting complexity of getting dependencies right for a TTK installation is overwhelming to users, and is thus a significant obstacle towards making TTK's methods available for a large user base.

3.1.1 Docker

For many scenarios (including scientific workloads), container technologies such as Docker [14] have proven to be a viable solution. In brief, a container captures not only an application, but also all of its dependencies in an otherwise self-contained, minimal install of a base operating system. Execution of containers is achieved via virtualization technologies built into all major operating systems. One goal of the packaging project was therefore to create a build process for Docker containers, which would expose TTK to a large user base.

The actual Docker packaging concept of TTK that was brought forth by the hackathon is based on the client-server model already provided by ParaView. Specifically, the docker container only contains a ParaView server build, and a full-option build of TTK, which is integrated into the ParaView server via its common plugin mechanism. Once this container is running, a vanilla ParaView client downloaded from Kitware's website can then be used to connect to the server running inside the container. Hence, all TTK algorithms are executed on the server within the container, and the results are send to the client.

During the hackathon, the workgroup was able to produce a minimum viable prototype. The discussions between hackathon participants were essential in deriving this server-client based container solution. It could be confirmed that the resulting containers are surprisingly easy to use and incur few limitations. Among the latter is a limitation to server-side software rendering, since hardware accelerated graphics within containers require complex vendor-specific setups that reduce the portability of the containers. Since the hackathon, the corresponding implementation underwent several improvements towards robustness and usability, and is now included in TTK. Currently, publicly available Docker containers for different ParaView versions can be downloaded from the DockerHub container registry [2].

3.1.2 Anaconda

The packaging workgroup also explored another approach to make TTK more accessible by providing TTK as an Anaconda [1] package. Anaconda is a cross-platform open-source distribution platform for Python-based data science software that includes its own package manager. Similar to a docker image, each Anaconda package specifies a build procedure and a list of dependencies; in this case to other Anaconda packages. VTK is already available as an Anaconda package, so the workgroup investigated if the same build structure can be adapted to the VTK layer of TTK. To this end, it was necessary to make use of custom CMake functions that are already included in VTK 9, but were undocumented at the time of the hackathon. The workgroup spend most of its time on including the various dependencies of TTK such as Sqlite, GraphViz, and Eigen. Based on the initial steps taken during the hackathon, the VTK-layer of TTK is now available as the Anaconda package TOPOLOGYTOOLKIT from the CONDA-FORGE channel [1].

3.2 Vector Field Robustness Module

Due to the limited availability of vector field topology algorithms in TTK, this workgroup set out to implement 2D vector field robustness calculation [19] and simplification [16] techniques.

3.2.1 Vector Field Robustness

In brief, robustness is a metric for pairing and canceling critical points using minimum perturbation of the L^∞ norm of vector magnitudes. The algorithm itself consist of three phases:

1. computation and classification of critical points, i.e., sources, sinks, and saddles;
2. construction of a specialized merge tree that considers the vector magnitude field and the previously calculated critical point locations and types; and
3. if the user intends to apply topological simplification, regions of the vector field—specified as sublevel set regions of the vector magnitude field—are numerically perturbed.

3.2.2 Implementation

The process of implementing this module can essentially be split into two parts: 1) building the module infrastructure, and 2) implementing the algorithm. The workgroup faced different struggles at each of these steps and required a lot of assistance from the organizers.

At the infrastructure level, there are multiple layers of code that need to be written, including the ParaView level, the VTK level, and finally the TTK level. Configuring these is non-trivial without extensive experience in at least VTK, but to some extent TTK as well. First, the selection and usage of data types at each level is non-trivial, especially if the module is supposed to support vector fields of different data types. Second, across all layers many definitions have to be repeated and consistent, which can easily result in errors, i.e., at the Paraview layer the user interface controls class members of the VTK layer, which are then passed to the base layer as parameters.

Implementing the algorithm portion of the module has its own mix of challenges. First is the used programming language. Fortunately, the original code was already written in c++, but if it would not have been, then the code would need to be translated first. Next, the code required translating existing mesh definitions into those used by TTK/VTK. Again, the workgroup members pointed out that it was fortunate that TTK's mesh specification is fairly easy to use, which made this code transition trivial. The final challenge was minimizing the use of external libraries, since ideally plugins should be self-contained. This was especially problematic for this particular module since the first phase of the algorithm—the computation and classification of critical points—was handled by an external library. Moreover, at the time of writing, TTK was only able to compute and classify critical points of scalar fields, so removing this dependency would require a lot of additional effort.

3.2.3 Results

The workgroup spend most of its time on the module infrastructure, leaving the actual implementation of the different phases of the algorithm to be completed after the hackathon. Yet, the participants felt confident to finish the module by themselves.

3.3 Extending the Integration of TTK in Inviwo

Inviwo [9] is a rapid prototyping framework for visualizing spatial and abstract data. The network editor of Inviwo provides the functionality for building a visualization pipeline out of individual blocks, or processors. A basic subset of TTK's functionality has been part of Inviwo for some time in form of various processors for creating TTK triangulations, topological simplification, and persistence diagrams. During the course of the hackathon, this subset was to be extended to also expose the Morse-Smale complex computation.

3.3.1 Implementation

There were some initial difficulties in understanding the extensive TTK infrastructure and the plethora of function arguments and results, which might partially have been caused by API requirements of VTK as TTK uses VTK to wrap algorithms. This is essentially the same limitation also reported by the vector field topology workgroup (subsection 3.2). These difficulties were, however, quickly resolved through interactions and discussions with the hackathon organizers. One design decision was to decouple the computation from the output required for the subsequent visualization, i.e. the geometric information of critical points and saddle point connections. Thus, two processors were created. The first accepts a TTK triangulation, performs the Morse-Smale calculations, and outputs the results. The second processor then generates geometric primitives from the results.

3.3.2 Results

At the end of the hackathon it was possible to compute the Morse-Smale Complex in Inviwo using the underlying TTK functionality and to visualize the results. Figure 4 depicts the Inviwo network computing the Morse-Smale complexes for a synthetic dataset. In the future, Inviwo will be extended to support an even larger part of the TTK functionality as well as incorporate periodic boundaries; a topic covered by another workgroup (see Section 3.4).

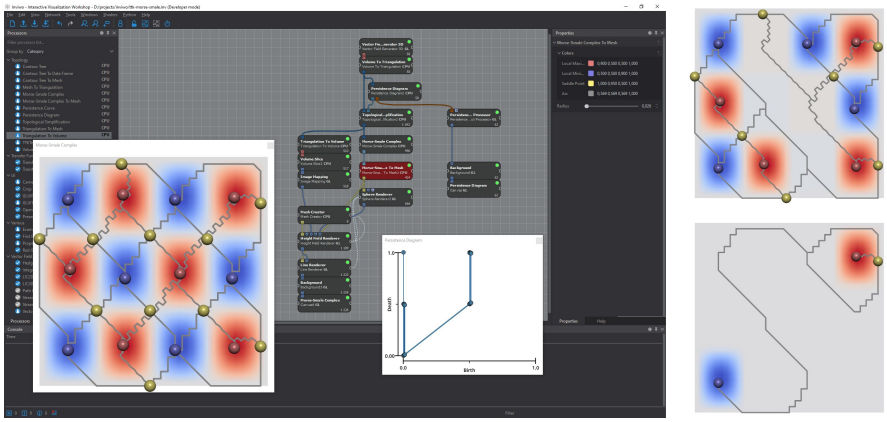

Fig. 4 Results of the TTK-based Morse-Smale complex computation in Inviwo. The figure shows the underlying visualization pipeline (background), and Morse-Smale complexes for different simplification thresholds (foreground/right).

3.4 Periodic Grids

Periodic boundary conditions are often used to model a very large (or infinite) system using a small representative part called a *unit cell*. Many physical, chemical, and biological systems exhibit repetitive symmetric patterns. These systems are ideal for study and analysis based on simulations on small unit cells with periodic boundary conditions. Other examples where periodic boundaries are used include the study of metals, crystal lattices, and bulk solvents in molecular dynamics simulations.

Topological algorithms that are already implemented in TTK—such as the Morse-Smale complex or merge tree computation—are able to process any domain that can be represented via a simplicial complex. However, TTK's original domain data structures—i.e., implicit and explicit triangulations—did not support periodicity. Therefore, this workgroup set out to extend the underlying triangulation data structures to support periodicity boundary conditions.

3.4.1 Implementation

At the base layer, TTK provides a unified triangulation interface that is capable of representing any domain, and this interface is used by all topological algorithms available in TTK. The TTK triangulation class structure is shown in Fig. 5. TTK distinguishes between two types of triangulations: `ExplicitTriangulation` to represent a triangulation where the cell connectivity is provided explicitly, and `ImplicitTriangulation` to provide a highly memory efficient triangulation for structured grids. `ImplicitTriangulation`, however, assumes that the boundary is not periodic.

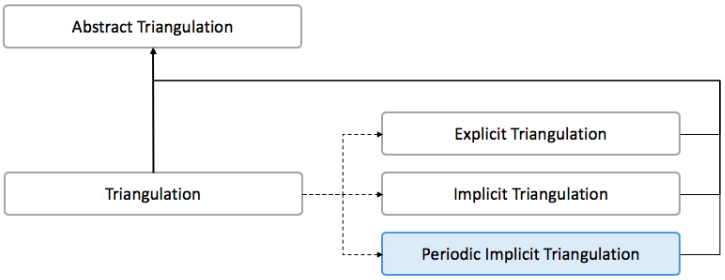

Fig. 5 The inheritance structure of the different triangulation classes in TTK.

The workgroup decided to implement a new triangulation class for structured grids with periodic boundaries called `PeriodicImplicitTriangulation`. Fig. 6 illustrates the concept behind this implementation for 2D grids, where additional edges and triangles are created between boundary vertices to establish periodicity in all coordinate directions. This concept directly translates to 3D grids. Based on the design decisions discussed during the hackathon, the workgroup completed the implementation of the new class after the hackathon and integrated it into the TTK master branch.

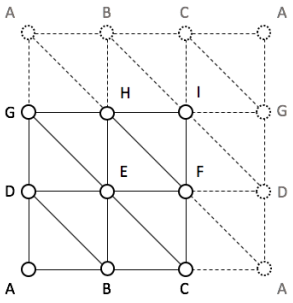

Fig. 6 The periodic triangulation in 2D is accomplished by adding the dashed edges and triangles to the triangulation shown with solid edges.

3.4.2 Results

A working example of the Morse-Smale complex computation on a periodic domain is shown in Figure 7. Currently, `PeriodicImplicitTriangulation` establishes periodic boundaries in all coordinate directions, but in the future this class can be extended to toggle periodicity for individual directions.

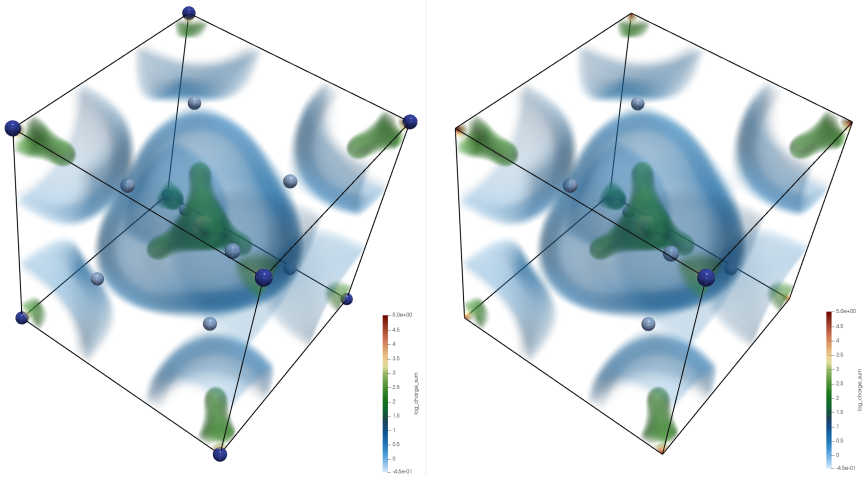

Fig. 7 Results of the Morse-Smale complex computation for a methane crystal modeled via a fixed unit cell that contains two methane molecules with non-periodic (left) and periodic (right) boundary conditions. The images show a volume rendering of the underlying scalar field and the location of the identified critical points, where maxima and saddles are represented as dark and light blue spheres, respectively. Note that without periodic boundaries eight maxima are obtained at the corners of the grid, however with periodic boundaries a single maximum is obtained.

4 Conclusion

Overall, the participants and organizers consider the hackathon a success, and the majority of attendees stated that they would participate again in a future hackathon. Besides the practical outcomes of the hackathon (Sec. 4.1), the senior developers also learned about organizing such an event (Sec. 4.2), and derived future development directions based on feedback of the participants (Sec. 4.3).

4.1 Workgroup Results

The hackathon initiated several significant coding projects that were later completed and added to TTK's source code. Specifically, due to the hackathon, TTK is now also shipped via a Docker image and an Anaconda package (Sec. 3.1), is further integrated into Inviwo (Sec. 3.3), and supports periodic grids (Sec. 3.4).

The coding project that worked on a new vector field robustness module (Sec. 3.2) is not yet completed, but yielded valuable insight into the practical challenges new developers face while adding new algorithms to TTK. Based on that insight, TTK's internal API is now getting revised (Sec. 4.3).

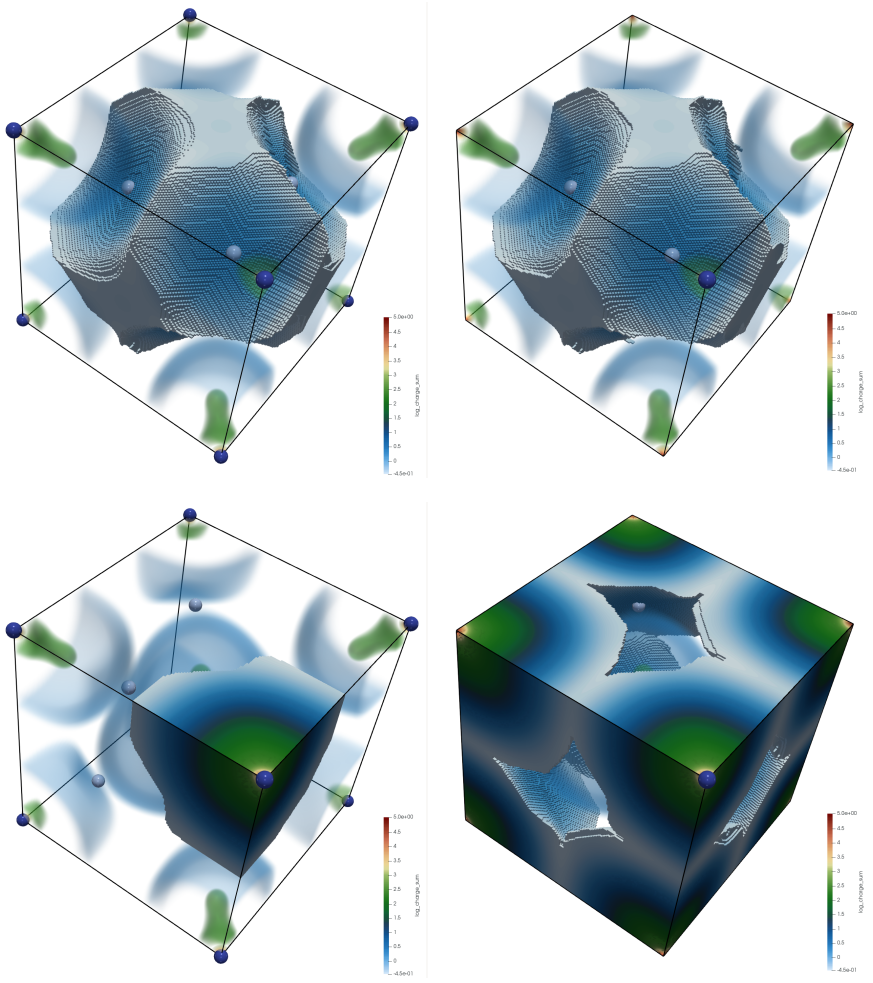

Fig. 8 Results of the Morse-Smale complex computation for a methane crystal modeled via a fixed unit cell that contains two methane molecules with a non-periodic (left) and a periodic (right) boundary. (First row) One methane molecule is located in the middle of the domain and hence it is correctly identified as an ascending manifold of a maximum in the Morse-Smale complex with or without periodic boundaries. (Second row) However, for the molecule located at the corner of the cell, the ascending manifold with periodic boundary correctly identifies the region corresponding to the second molecule. The ascending manifold without periodic boundary fails to correctly segment the volume into two methane molecules.

4.2 Organizational Aspects

A major factor for the success of the TTK hackathon was to organize it as a co-located event at the TopoInVis conference. This was advantageous to both the hackathon organizers as well as the participants, as conference participants correspond to TTK's target user base, and participants were able to simply extend their stay by one day to attend the hackathon.

Although the organizers originally intended to spend the same amount of time on introducing TTK and working on the actual coding projects, it turned out to be better to spend more time on the actual coding sessions. In future hackathons, the organizers recommend to use the revised schedule (Fig. 3), and to adjust the content of the introduction session based on the attendees. To this end, the initial questionnaire prior to the hackathon was essential in assessing the different experience levels and interests of the participants. A shortcoming of the organization was the missed opportunity to collect a formal feedback of the participants via another questionnaire. The organizers strongly recommend to do so in future events.

4.3 TTK Development Directions

Already at the beginning of the hackathon it became apparent that many participants struggled with installing TTK on their system. In fact, many potential users are currently discouraged to even try out TTK because it is very difficult to get it up and running. The hackathon sprout two valuable approaches to this problem by making TTK accessible via a Docker container and an Anaconda package. Yet, the Docker container requires superuser privileges and only supports software rendering, and the Anaconda package does not feature ParaView as a front end. Besides advancing these approaches, it also seems valuable to improve and simplify the general build process of TTK in the future.

Moreover, it appeared that TTK's internal API could be greatly simplified in the interest of readability and accessibility to new developers, without changes in performance or in its core design principles. Based on feedback from the participants, the senior developers are currently revising TTK's internal API to make the code base more transparent, simpler, and easier to modify, in order to drastically reduce the amount of effort users and developers have to spend on learning and extending TTK.

In conclusion, it is the belief of the organizers that the hackathon was essential for growing TTK's user and developer base, for addressing current limitations, and deriving future development directions.

Acknowledgements We thank all participants for their significant contributions and valuable feedback. This work was partially supported by the European Commission grant ERC-2019-COG "TORI" (ref. 863464), and the German research foundation (DFG) within the IRTG 2057 "Physical Modeling for Virtual Manufacturing Systems and Processes".

References

1. Anaconda Software Distribution, 2016. https://anaconda.com.
2. TTK Docker Container Repository, 2019. https://hub.docker.com/u/topologytoolkit.
3. Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.
4. Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological Persistence and Simplification. *Discrete & Computational Geometry*, 28(4):511–533, 2002.
5. Martin Falk, Christoph Garth, Charles Gueunet, Joshua Levine, Jonas Lukasczyk, Julien Tierny, and Jules Vidal. Topological data analysis made easy with the topology toolkit, a sequel. 2019. `https://topology-tool-kit.github.io/ieeeVisTutorial.html`.
6. Guillaume Favelier, Charles Gueunet, Attila Gyulassy, Julien Kitware, Joshua Levine, Jonas Lukasczyk, Daisuke Sakurai, Maxime Soler, Julien Tierny, Will Usher, et al. Topological data analysis made easy with the topology toolkit. 2018. `https://topology-tool-kit.github.io/ieeeVisTutorial.html`.
7. Charles Gueunet, Pierre Fortin, Julien Jomier, and Julien Tierny. Task-Based Augmented Merge Trees with Fibonacci Heaps. In *IEEE Symposium on Large Data Analysis and Visualization*, 2017.
8. Charles Gueunet, Pierre Fortin, Julien Jomier, and Julien Tierny. Task-Based Augmented Reeb Graphs with Dynamic ST-Trees. 2019.
9. Daniel Jönsson, Peter Steneteg, Erik Sundén, Rickard Englund, Sathish Kottravel, Martin Falk, Anders Ynnerman, Ingrid Hotz, and Timo Ropinski. Inviwo-a visualization system with usage abstraction levels. *IEEE transactions on visualization and computer graphics*, 2019.
10. Pavol Klacansky, Julien Tierny, Hamish Carr, and Zhao Geng. Fast and exact fiber surfaces for tetrahedral meshes. *IEEE transactions on visualization and computer graphics*, 23(7):1782–1795, 2016.
11. Jonas Lukasczyk, Christoph Garth, Gunther WeberWeber, Tim Biedert, Ross Maciejewski, and Heike Leitte. Dynamic nested tracking graphs. *IEEE transactions on visualization and computer graphics*, 2019.
12. Jonas Lukasczyk, Eric Kinner, James Ahrens, Heike Leitte, and Christoph Garth. VOIDGA: A View-Approximation Oriented Image Database Generation Approach. In *IEEE 8th Symposium on Large Data Analysis and Visualization (LDAV)*, 2018.
13. Jonas Lukasczyk, Gunther Weber, Ross Maciejewski, Christoph Garth, and Heike Leitte. Nested tracking graphs. In *Computer Graphics Forum*, volume 36, pages 12–22, 2017.
14. Pankaj Saha, Angel Beltre, Piotr Uminski, and Madhusudhan Govindaraju. Evaluation of Docker Containers for Scientific Workloads in the Cloud. *CoRR*, abs/1905.08415, 2019.
15. Nithin Shivashankar and Vijay Natarajan. Parallel Computation of 3D Morse-Smale Complexes. *Computer Graphics Forum*, 31:965–974, 2012.
16. Primoz Skraba, Bei Wang, Guoning Chen, and Paul Rosen. 2D Vector Field Simplification Based on Robustness. In *IEEE Pacific Visualization Symposium (PacificVis)*, pages 49–56. IEEE, 2014.
17. Julien Tierny, Guillaume Favelier, Joshua A Levine, Charles Gueunet, and Michael Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics*, 2017. `https://topology-tool-kit.github.io/`.
18. Julien Tierny and Valerio Pascucci. Generalized Topological Simplification of Scalar Fields on Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2005–2013, 2012.
19. Bei Wang, Paul Rosen, Primoz Skraba, Harsh Bhatia, and Valerio Pascucci. Visualizing Robustness of Critical Points for 2D Time-Varying Vector Fields. In *Computer Graphics Forum*, volume 32, pages 221–230. Blackwell Publishing Ltd Oxford, UK, 2013.