ISSN 2186-7437

NII Shonan Meeting Report

No. 193

The Moving Target of Visualization Software – Closing the Gap between Research and Application

Christina Gillmann
Takayuki Itoh
Michael Krone
Alexander Lex
Guido Reina

February 12–16, 2024

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

The Moving Target of Visualization Software – Closing the Gap between Research and Application

Organizers:
Christina Gillmann (University of Leipzig, Germany)
Takayuki Itoh (Ochanomizu University, Japan)
Michael Krone (University of Tübingen, Germany)
Alexander Lex (University of Utah, USA)
Guido Reina (University of Stuttgart, Germany)

February 12–16, 2024

Abstract

Visualization, having matured within computer science, is now recognized as a vital tool for data analysis across various fields. Its widespread adoption spans scientific research and extends to industry applications, exemplified by platforms like Microsoft PowerBI and Tableau. Moreover, visualization is a pivotal means of conveying complex data topics in the news media, such as COVID-19 spread or political poll uncertainty. Despite the proliferation of tools and methods developed by the academic visualization community, many promising research prototypes fail to attain broad adoption due to issues of sustainability and extensibility, often tied to the limited lifespan inherent in the academic career cycle. Compounded by the rapidly evolving software ecosystem, which includes shifts in execution environments, programming languages, and interaction paradigms, PhD students face significant challenges in maintaining and advancing their research prototypes amidst their academic pursuits. Recognizing the importance of this issue to our community, we have organized a series of events to address pressing concerns and propose solutions to enhance the practical aspects and research efficacy within our field. The inaugural event, Shonan Seminar #145 (February 2019), convened industry, national labs, and academia participants, resulting in the identification of nine key concerns and opportunities, subsequently documented in a publication [22] that delineates the current state and future directions of our community.

Background and introduction

Visualization has evolved into a mature subfield of computer science. It has become widely accepted as an essential data analysis method in diverse fields. Visualization enables scientific data analysis and is widely adopted in industry, as the success of business data visualization platforms such as Microsoft PowerBI or Tableau demonstrates. Visualization is also widely used in the news media to communicate data about topics as diverse as the spread of COVID-19 or the uncertainty associated with political polls. Over the years, the academic visualization community has developed many tools and methods, many of which have been widely adopted in diverse application domains. However, many research prototypes never reach the maturity necessary for broad adoption, even though the underlying method has significant merit. These prototypes are often neither sustainable nor easily extensible for subsequent research projects, and their lifespans are often tied to the original author's academic career. Since Ph.D. students usually implement research prototypes in academia as part of their thesis projects, this results in a rather short lifespan. Another aspect that adds to this problem is that the whole ecosystem around software quickly evolves. This, for example, includes changes in the execution environment (software as well as hardware), preferred programming languages, external third-party libraries, and interaction paradigms. Consequently, Ph.D. students would have to invest a lot of time and effort to keep up with the moving target of developing and maintaining usable software while also doing actual research at the same time. We found this topic to be of great interest to many members of our community, which is the reason why we have recently organized several related events to discuss the most pressing issues and propose medium-term solutions that can improve both the practical aspects of our daily work and the quality and efficiency of our research contributions. The first event in this series was the Shonan Seminar # 145 (February 2019, organized by H. Childs, T. Itoh, M. Krone, and G. Reina), in which the group (consisting of 24 participants from industry, national labs, and academia) distilled the nine most interesting concerns and opportunities. We have meanwhile transformed these results into a publication that shows the state of our community as well as possible future directions [22] and followed up with additional events and venues, like the establishment of the EuroVis Workshop VisGap, which has been successful since 2020.

Overview of the meeting

In this seminar, we wanted to bring together leading visualization researchers and practitioners to discuss the specific challenges around visualization software. We identified the following challenges we want to work on during this seminar. Visualization research usually requires an implementation of the proposed method or approach for evaluation. However, the visualization community lacks incentives to publish research software, much less evolving a research prototype into a usable tool. Furthermore, especially young researchers usually lack the training and experience to develop sustainable research software. As mentioned above, the requirements for novel approaches are steadily increasing due to data scale and complexity, as well as the increasing maturity of the field itself and the influences from other areas, such as machine learning or human-computer interaction. Consequently, the prototypes usually are not easily extensible for subsequent research projects. In order to close the gap between research and application, we want to develop proposals and strategic initiatives to solve these challenges and to move the whole community forward.

The proposed Shonan Seminar aimed to intensify and advance the discussions started at Shonan Seminar #145 by elaborating on the unfinished discussions and examining additional topics. One new aspect beyond the previous seminar was the discussion of concrete models for incentivizing visualization software within the research community (e.g., tools papers, which are common and considered essential contributions in other scientific disciplines, mandatory software accompanying submissions, etc.). In addition, we aimed to specify requirements and changes in the VIS community that need to be fulfilled in order to ease the translation of research prototypes into applicable software.

Overview of Talks

Uncertainty-aware visual analytics and transferability

Christina Gillmann, University of Leipzig, Germany

The talk briefly introduced the concept of uncertainty-aware visual analytics and the problems while bringing this into a software. The attempts to do so where mainly framed around the ParaView ecosystem that is on the one hand flexible, but on the other hand documented insufficiently. Early algorithms have been integrated into open source software like ITK and shared projects on GitHub. This was followed by a list of future initiatives that are planned in the research agenda such as applying for proper funding. Finally, the talk highlighted the 3 top challenges in open source visualization software that are the acquisition of proper funding, generalization/standards and accessibility.

Making Sense of our Software Ecosystem and Sustaining it

Guido Reina, University of Stuttgart, Germany

The talk reflected briefly on the richness of the visualization software ecosystem, but also criticized that we are not properly taking advantage of it. This is caused by several challenges, not the least of which is the diversity of technical platforms that are used. We have only been partially successful at creating software that facilitates composition of more complex approaches from either minimally coupled building blocks or agreeing on an overarching framework or platform we can all make use of. There also is no proper schema or other formalization that can capture the capabilities and technical implications of the existing software, so especially fresh Ph.D. candidates have little to go on when starting a new project. Finally, sustaining our software in a way that makes it relevant to practitioners, especially from industry, is an ongoing challenge.

Visualization and Visual Analytics Tools for Biomedical Application Cases

Michael Krone, University of Tübingen, Germany

In my talk, I briefly introduced myself and showed examples of methods and tools for the visual analysis of biomedical data that I worked on in the past. This specifically includes the interactive visualization of large, dynamic molecular structures from simulations, visual analytics tools for biomolecular data in the context of biological pathways, and intensive care unit data visualizations. These applications were either realized within the open-source visualization frameworks MegaMol [10] and VMD, using game engines, or as web applications. These platforms allowed for effective rapid prototyping of novel visualization tools. From my point of view, balancing the time spent on software development and research is an important challenge, especially for small research groups. However, building usable visualization research tools is also important for collaborations with users. This could be partially incentivized by establishing possibilities to publish tools papers, as is common in other communities like bioinformatics.

Technical Scientific Transfer for Visualization Software

Michael Keckeisen, TWT GmbH Science & Innovation, Germany

In this talk, I explored the significance and complexities surrounding the transfer of scientific knowledge into practical industrial applications, with a focus on visualization software. Addressing sectors such as automotive, aerospace, energy, and healthcare, I discussed challenges like user-centric automated personalization of data visualizations, explainable artificial intelligence through visualization, and the collaborative examination of big data and multidimensional data through coupled augmented reality visualizations.

TWT GmbH Science & Innovation is specialized in technical-scientific transfer. At the interface of informatics, engineering and mathematics, TWT works directly with customers and partners such as Mercedes-Benz, DaimlerTruck, BMW, Audi, Porsche, CARIAD, IBM, Bosch, ESA, Airbus and Samsung.

An example of TWT's scientifc contributions is the GETMe mesh smoothing method [25].

Diversity of users and platforms

Takayuki Itoh, Ochanomizu University, Japan

This talk first introduced the representative studies of the speaker including fast isosurface generation, Treemap-like hierarchical data visualization, graph visualization, and multidimensional data visualization. Then, the speaker mentioned the top three open problems as follows:
(1) Addressing novice to expert users by a single software. Preferable visualizations may be different between novice and expert users. Satisfaction of both types of users will make visualization software more robust and general-purpose.
(2) Not only users, but also platforms should be wider. Support of a wide range of platforms (mobile, VR/AR, ...) by a single software will also be a key point of visualization software.
(3) From an aspect of the visualization research community, the discipline of visualization software development is also an important issue. It should be more well-defined regarding what is good research, and what are relevant papers contributions.

Sustainable FAIR VIS Software

Christoph Garth, RPTU Kaiserslautern-Landau, Germany

The talk focuses on two fundamental questions related to visualization software: 1) How can visualization software adhere to the FAIR principles, which are rapidly becoming a requirement that researchers have to adhere to obtain funding and publish; here, especially reusability and reproducability are aspects that are intrinsic to the VIS community. While containerization and other techniques increase both, they are not a direct step toward becoming more FAIR. Other FAIR aspects, such as metadata annotation and accessibility, have not been considered to great extent within the community. 2) How can software libraries be developed and maintained within the community? The open-source Topology Toolkit (TTK) is a successful example of a software that serves as

a basis for research in the domain of topological data analysis, with a small but active community maintaining and developing it. How can such activities be nurtured and strengthend? It appears that increasing the academic reward of software activities, e.g. through software publications (in analogy to data publications), could be beneficial here.

Visualization of Big Complex Networks

Seok-Hee Hong, The University of Sydney, Australia

This talk briefly reviews the state of the art for big complex graph visualization, including the sublinear-time algorithms to address the scalability, as well as *faithful* graph drawing to show the ground truth structure of complex graphs.

Based on my experience as a project leader to develop a visual analytic tool GEOMI for large graph visualization and collaboration with industry and domain experts in systems biology and social networks, I present the following top three open problems for visualization software: (1) closing the gap between theory (research) and practice (tool); (2) designing a unified open source platform, integrating analysis and visualization; (3) supporting different users, such as visualization researchers, data scientists, students, and domain experts.

David Laidlaw's Top Open Problem Examples

David H. Laidlaw, Brown University, United States

Laidlaw identified several challenges regarding software for visualization. The first was the absence of a mechanism for transferring novel visualization approaches to potential users. Kitware might serve that purpose, but it could be more effective. The second challenge is the cost of creating virtual-reality visualization applications for domain science exploratory data visualizations. Typically these can take many months or years for each example. Software that speeds this development process could greatly accelerate scientific progress. A third challenge is the lack of clarity around open research challenges in visualization. A clearer agenda could help focus research and also provide input to funding agencies about where they might best make investments. Laidlaw is known for painting-motivated multi-valued visualizations, virtual reality visualization of bat flight and dinosaur footprint formation, diffusion MRI visualization and analysis, and a room-sized retinal-resolution virtual reality display known as the yurt. He worked on many software projects since 1989 when he was one of the creators of the dataflow visualization system AVS. He is also adept at building campfires and stone walls.

Visual Exploration Tools for Single-Cell Biology

Fritz Lekschas, Ozette Technologies, United States

Fritz Lekschas presented current visual exploration tools for exploring large-scale single-cell data and discussed challenges and opportunities in bringing these software tools closer to the biologists. One such tool, developed at Ozette Technologies, is a scalable embedding visualization for single-cell data that

provides a high-resolution overview of the immune cell landscape. Implemented originally as a purely web-based visualization, Lekschas' team further improved the tool in two ways. First, they modularized the visualization tool such that it can be integrated more easily into other software. Second, they added support for running the tool as an interactive widget in Jupyter Notebook/Lab to directly integrate into the biologists workflow. And third, they integrated the tool's APIs into the Python data ecosystem for simplified re-use with other datasets. Fritz Lekschas concluded his talk arguing that similar modularization and standardization approaches could prove useful for fostering a wider adoption of visualization software by the data science community.

Visualization of Massive Scientific Data

James Ahrens, Los Alamos National Laboratory, United States

James Ahrens described the state of visualization tools for visualizing massive scientific data. Ahrens is the founder and design lead of ParaView, a widely used, open-source, visualization tool for massive data. Ahrens identified future challenges for scientific visualization of massive data. The first is the recognition that science is informed by ensembles of experimental and simulation data. Existing visualization tools can handle a single massive simulation or experiment including a time series results and multiple variables but he asked what about thousands of these? Ahrens noted ensembles are used extensively for machine learning applications. Ahrens identified the need to capture, represent, process and visualize ensemble metadata and data. A second challenge he identified is the integration of machine learning into the visualization process as well as the need for the visualization community to help visually understand machine learning results. A third challenge was an idea to re-explore the fundamental data representation for visualization to support seamlessly transitioning from 1D, 2D, 3D to ND visual representations.

Software Development for Biomedical Image Informatics - Examples and Challenges

Katja Bühler, VRVis, Vienna, Austria

Katja Bühler gave an overview of the field of biomedical image informatics, including AI-based image analysis and semantic enhancement, visualization and integration of image data, data integration and integrated analysis of image data across multi-modal imaging and spatial *omics data, and related software projects. As an example, she presented the Brain* system, which provides spatial integration, rapid interactive access, visualization, mining, and analysis capabilities for massive collections of neuroscience data. She concluded with a discussion of one scientific and two community challenges related to visualization software: (1) How will AI affect the field in the future? (2) How can we provide a sustainable software base for research and development that reflects the state of the art? (3) How can we build a community that works together on state-of-the-art visualization software for the common good?

Challenges in Software Development for Interactive Exploration of Simulation Ensembles

Kresimir Matkovic, VRVis, Vienna, Austria

Kresimir Matkovic gave an overview of the practice of interactive visual analysis within the realm of simulation ensembles, as well as of the practice of handling complex data in diverse domains such as engineering, medicine, geology, and ergonomics. He highlighted three key challenges for developers in modern visualization software: **(1.) Rapidly Evolving Technologies:** Keeping up with the fast-paced changes in new (web) technologies is a significant hurdle. This involves making the right choices amid constant changes and finding skilled developers familiar with these evolving tools. While the abundance of software and libraries spurs innovation, it poses the challenge of selecting the most effective tools. **(2.) Integration of Research Ideas:** Many excellent ideas from research in visualization remain confined to academic circles and aren't seamlessly incorporated into standard software. The limited awareness and involvement of developers outside the visualization research community worsen this problem. Bridging this gap is essential to unlock the full potential of these innovative concepts and encourage collaboration between researchers and developers from diverse backgrounds. **(3.) AI for Visualization (AI4Vis):** The rise of AI in visualization presents both a formidable challenge and a significant opportunity. The key question is whether we will effectively use AI to advance our capabilities. Successfully navigating complexities, fostering collaboration, and embracing AI's transformative potential are crucial steps toward enhancing and democratizing the field of visualization.

Visual Exploration of Software and Systems

Katherine (Kate) E. Isaacs, The University of Utah, United States

Katherine (Kate) Isaacs presented an overview of visualization and visualization software needs for analyzing software and computing systems. She emphasized the need to meet users where they are, showing examples of embedding visualizations in both computational notebook and command line interfaces. She also expressed maintenance issues with visualization software, noting they even arise quickly in projects where the domain team were the main developers of the visualization. While she noted the strengths of current visualization software for providing first pass solutions and for prototyping, she noted these tools do not yet provide good support for charts commonly used in software and systems visualization, such as interconnected timelines, especially at the scale of the data these domains generate.

Audience-targeted Visualization Design for Optimal Usability

Kwan-Liu Ma, University of California at Davis, United States

Developing a visualization with maximum usability must take into account the purpose of the visualization, the target users, the characteristics of the

data, and the viewing device and environment. Understanding the users is the first and the most important task to perform. This task may be done through extensive interviews, which is then followed by an iterative process of design-prototyping-evaluation with the users closely engaged. I show how to design scientific visualization at extreme-scale that is audience-targeted, task-directed or physically-based, and both computationally and visually scalable for optimal usability [23, 20, 27]. The resulting visualization can enable scientists to effectively validate their data, uncover previously unseen features, and communicate their work to others.

The Changing Landscape for Visualization Tools

Cláudio T. Silva, New York University, United States

Over the last two decades, visualization techniques and tools have matured and are widely used. Many tools have been developed as powerful standalone tools (e.g., ParaView and OpenSpace). While useful and flexible, such comprehensive tools tend to be fairly complex, and require a steep learning curve on users. Alternatively, the visualization community has also developed simpler tools, tailored to particular domains, (e.g., BirdVis and TaxiVis). While easier to use for their intended purpose, these tools are usually not as flexible, and require users to transition to other approaches as their needs evolve and shift. In this short talk, I discussed how new approaches based on designing "visualization languages" and "notebook" interfaces might provide a way for users to evolve their tools more easily and naturally as their needs change.

Software for HPC Scientific Visualization

Kenneth Moreland, Oak Ridge National Laboratory, United States

In this talk I briefly review a history of research to enable large-scale scientific visualization on high-performance computing (HPC) machinery. The work starts with parallel rendering to push large polygon structures to rendered images on large-format displays. The following work moved to fully-featured scientific visualization by participating in the ParaView application. This software has been a principal component in delivering visualization research to end users. More recent work expands the implementations to use modern accelerator processors for visualization processing.

Although creating usable software does add a significant burden additionally over base research work, such work is beneficial not just to end users but to researchers themselves. In reviewing the number of references to my past publications, the references are more correlated to the availability of software described than the strength of the novel contribution. The lower the barrier to using software, the more likely other researchers will use that to support their research or build upon your results.

Connecting Visualization Research and Domain Research

Marc Baaden, CNRS Paris, France

It is a challenge to pass on the latest advances in visualization research

to specialist scientists. As the head of a molecular modeling and theoretical biochemistry lab, I face this obstacle on a daily basis and have to deal with practical problems that require a solution. My involvement revolves around directing research efforts towards molecular graphics, interactive simulations and virtual reality applications, all centered around the development and maintenance of the UnityMol software.

Using UnityMol, we are exploring novel representations such as HyperBalls, evaluating the feasibility of using a game engine for development and prototyping purposes, and striving to facilitate FAIR exchange of visualization experiences. In my area of work, the pure elements of visualization merge seamlessly with augmented reality, virtual reality and tangible IoT objects.

In the future, we will explore multi-user collaboration, address the dynamics of consciousness and presence, incorporate explainable AI into molecular visualization, explore VR in cloud environments, and adapt to the challenges of large data sets and simulation ensembles.

In my specific subfield, the strengths of the visualization software lie in its extensive range of functions, its robust code base and its mature stability. However, interoperability, the lack of metadata for visualization objects, insufficient integration with augmented reality technologies and the lack of multi-user standards and functionalities are areas for improvement.

Solving the most important unsolved problems requires the creation of a sustainable "business model" for academic software. In addition, developments need to be seamlessly integrated into data practices to promote FAIR principles and open science amidst the data deluge. Linking these developments to application domains remains a critical frontier in our quest for progress.

Broaden participation and create composable software

Dominik Moritz, CMU and Apple, United States

In this introduction talk, Dominik argues that we still have a long way to go to broaden the diversity of who participates in software development. While this challenge is not unique to visualization software we should do our part in creating a welcoming environment for everyone to publish their visualization software or contribute to existing projects.

Dominik also presented an overview of the visualization libraries and frameworks he and his groups at CMU and Apple developed (e.g., Mosaic). He argues for creating not just *usable* but *composable* software. By building small building blocks rather than monoliths, we are more productive with less code, enable better reuse and comparison, and make our software more useful. To get to composable software, we need to value API design and make it part of our research contributions.

Translating Research into Practice

Anamaria (Ana) Crisan, Tableau Research, United States

This talk briefly discussed existing challenges of translating visualization research and software into industrial products. I highlight several challenges. The first is the difficulty of integrating visualization research into existing analytic

workflows. The integration with commercial workflows can be more challenging and code artifacts can be difficult to translate directly; opportunities for integrating with open sources tools, existing, for example, with Python and R, are more fruitful, but still not regularly done. I also highlight alignment challenges between graduate student incentives and training and the requirements of industry. Finally, I describe the unique regulatory, legal, and legacy environments in industrial enterprises and how this can hinder the research translation process.

Visualization Software Development for Scientific Workflows

Gerik Scheuermann, Leipzig University, Germany

The talk briefly covered several works on feature and topology-based visualization, mainly tailored towards fluid dynamics, mechanical engineering, and medical application domains. It also covered several software systems like the Field Analysis with Topological Methods (FAnToM) software for feature and topology based visualizations of scalar, vector, and tensor fields[1] [26]. In addition, the software package OpenWalnut for the visual analysis of multimodal brain imaging data like DTI, EEG, MEG, HARDI, fMRI, etc.[2] [6] Furthermore, we presented the SARDINE tool for integrating NLP-processed legal documents, online sensor data, publicly available online resources like weather forecasts, LIDAR measurements by drones in a GIS-like system for regional planning purposes[3] [1]. Another software tool named LexCube was explained. It shows data cubes from geo-sciences in an interactive fashion in the browser, but also as part of a Jupyter notebook with an interactive 3D window[4] [24]. The tool also allows for a physicalization by printing out some cube on a piece of paper that can be folded into a physical data cube.

The talk finished with some of the challenges that we recognized during the development of those tools which include how to integrate visualization software into the workflow of users. This works best, based on our experience, if the application domain scientists or engineers show a strong interest in the software and come up with the plan of developing such a tool themselves. Then, we as a visualization group join this effort with our visualization and software development expertise. In this case, our group supports and supported such developments in LexCube, OpenWalnut, and GeoTemCo[5] [15]. Current experience with this approach is also drawn from showing molecular dynamics simulations in biochemistry by a web-based tool called MDserv[6] [17].

Strategies for Scientific Software Engineering

Alexander Lex, University of Utah, United States

The visualization community has historically focused on large, monolithic, interactive visualization systems. However, experience has shown that these

[1] http://www.informatik.uni-leipzig.de/fantom/
[2] https://openwalnut.org/
[3] https://www.uni-leipzig.de/newsdetail/artikel/projekt-sardine-soll-nachnutzung-von-braunkohletagebauen-erleichtern-2022-06-30 (only in German)
[4] https://www.lexcube.org/ and https://pypi.org/project/lexcube/
[5] http://www.informatik.uni-leipzig.de/geotemco/
[6] https://proteinformatics.informatik.uni-leipzig.de/mdsrv

systems are hard to maintain and are rarely adopted outside of a narrow set of users. In my talk, I pose the question on whether building such systems is futile in the first place. I argue that systems of similar complexity that are developed by commercial entities are usually staffed with dozens or even hundreds of engineers, product managers, UI/UX designers, etc. As a remedy, I propose that the community focus on smaller, easier to maintain reusable components, and provide maintenance and documentation for these. There are several prominent libraries that are developed and maintained in this way, such as D3, Vega, or Matplotlib. While there are challenges with this approach, such as limited interactivity, these are interesting research topics that the community can tackle.

In Situ Visualization Tools and Beyond

Gunther H. Weber, Lawrence Berkeley National Laboratory, United States

In my talk I briefly reviewed my previous work on visualization of adaptive mesh refinement (AMR) data, domain specific visualization tools, topology-based visualization and parallel data analysis and visualization on high performance computing systems. Furthermore, I discussed recent work for the Exascale Computing Project (ECP)—including the integration of algorithms and visualization approaches into VTK-m and Ascent—and the Scalable In Situ Analysis and Visualization (SENSEI) project. In the future, one of my goals is to develop visualization methods and software that help understand scientific machine learning models. Furthermore, I plan to to develop new approaches at the edge for experimental and observational facilities and for automated laboratories.

In my previous work, I found that visualization tools like VisIt and frameworks simplify development of new visualization methods and deploying them to wide audience. Furthermore, there is a growing number of software frameworks in many languages (C++, Python, JavaScript) that simplify building new visualization software. However, many challenges remain. Deploying visualization tools not based on widely available frameworks is often difficult. Securing funding and "hands on keyboard" to maintain developed software is also often difficult. Finally, many available frameworks (e.g., Open3D in Python) are often targeted to non-visualization applications and using them for visualization is often not ideal.

I see the following (research) challenges for the development of visualization software: (i) Maintaining visualization software research prototypes in a quickly changing deployment environment (both funding and reproducibility), (ii) Combining web-based front-ends with HPC back-ends, (iii) Distributed visualization systems incorporating and synchronizing many mobile devices, (iv) Increasing the modularity of frameworks (for lightweight linking), and (v) Targeting wider range of accelerators/hardware (FPGA, neuromorphic computing).

Visualization in the application domains and their software development.

Daniel Wiegreffe, Leipzig University, Germany

The talk briefly introduced different visualization systems for exploring

biological data and spatial data. Various aspects of software development were then discussed, based on the experience gained in the development of these systems. It was emphasized that current technologies have made it much easier to develop platform-independent solutions easily. Nevertheless, there are also challenges, such as the availability and maintenance of software projects. An additional topic of discussion was the need for better interoperability between different software components. In addition, the connection between software development and research in the field of visualization was discussed and how the sustainability of software packages in this field can be improved.

OpenSpace — Astronomy Software for the masses

Alexander Bock, Linköping University, Sweden

This talk focused on the OpenSpace platform, which is an open-source software package originating from the visualization domain and which is designed to visualize the entire known universe. It is an environment for novel visualization research, a research tool for astronomers, as well as a tool for the public dissemination of astronomy discoveries. The latter usage is mainly targeted at interactive planetarium venues. Usage of the same software in all these domains enables the rapid deployment of research results to the domain scientists and the general public without the need of format conversions when transitioning from a "research software" and thus enables a short-circuiting of the knowledge dissemination pipeline.

This software is based on a second, also open-sourced library called Simple Graphics Cluster Toolkit (SGCT) that provides a backend for porting arbitrary graphics application into immersive environments such as stereoscopic dome theaters, CAVEs, and others.

Visualization Software for Cellular Mesoscale

Ivan Viola, KAUST, Saudi Arabia

The technical aspect presented in the talk were related to MesoCraft, a procedural modeling platform in which cellular mesoscale structures can be formulated through a set of spatial relationships between individual molecular building blocks. This software has showcased a promising new direction where development is performed in C++ with WebGPU as the graphics API, which allows the software to be deployed as a desktop application on all OS platforms as well as a web application. The long-standing desire for single code base compiled either as a local executable or as a client-side web application has become a reality. The web application secures an easy deployment for any user, and it also allows for additional benefits, such as user management, the possibility of storing cellular mesoscale models in a central repository that can be shared among bio-science researchers who together complete a cellular mesoscale model of an underlying sub-micron biological system.

List of Participants

- Christina Gillmann, University of Leipzig
- Takayuki Itoh, Ochanomizu Univeristy
- Michael Krone, University of Tübingen
- Alexander Lex, University of Utah
- Guido Reina, University of Stuttgart
- James Ahrens, Los Alamos National Laboratory
- Marc Baaden, CNRS, Institut de Biologie Physico-Chimique
- Alexander Bock, Linköping University
- Katja Bühler, VRVis
- Anamaria Crisan, Tableau Research
- Christoph Garth, University of Kaiserslautern-Landau
- Hans Hagen, Technical University Kaiserslautern (RPTU)
- Seok-Hee Hong, University of Sydney
- Katherine Isaacs, The University of Utah
- Michael Keckeisen, TWT GmbH Science & Innovation
- David Laidlaw, Brown University
- Steve Legensky, Intelligent Light
- Fritz Lekschas, Ozette Technologies
- Kwan-Liu Ma, University of Californa Davis
- Kresimir Matkovic, VRVis / University of Zagreb
- Kenneth Moreland, Oak Ridge National Laboratory
- Dominik Moritz, Carnegie Mellon University
- Gerik Scheuermann, Leipzig University
- Claudio Silva, New York University
- Ivan Viola, KAUST
- Gunther H. Weber, Lawrence Berkeley National Laboratory
- Daniel Wiegreffe, Leipzig University

Meeting Schedule

Check-in Day: February 11 (Sun)

- Welcome Banquet (19:00-20:30)

Day 1: February 12 (Mon)

- Opening (9:00-9:30)
- Self introduction talk (9:30-12:00, 13:30-14:30)
- Problem definition (15:00-18:00)

Day 2: February 13 (Tue)

- Discussions (9:30-12:00, 13:30-18:00)
- Group Photo Taking (12:00)

Day 3: February 14 (Wed)

- Discussions (9:30-12:00)
- Excursion and Main Banquet (13:30-21:00)

Day 4: February 15 (Thu)

- Discussions (9:30-12:00, 13:30-16:30)
- Demo (16:30-18:00)

Day 5: February 16 (Fri)

- Report writing (9:00-11:00)
- Wrap up (11:00-11:30)

Summary of discussions

The following section will present a summary of the discussions for each topic that has been identified as important in recurrent plenary discussions that would adjust this ranking based on participants' interests, coverage, and previous discussions.

Topic: User interface design, user experience design, and accessibility of vis software

Participants: Anamaria Crisan, Alex Lex, Ivan Viola, Kwan-Liu Ma, Seok-Hee Hong, Frtiz Lekschas, Steve Legensky, Takayuki Itoh

Graphical user interfaces (GUIs) serve as the bridge between the user and the complex world of data. Their primary purpose is to facilitate effective interaction, interpretation, and manipulation of data, enabling users to derive meaningful insights. Throughout our discussion sessions, we delved into the need for continuous adaptation of user interfaces to meet both user requirements and the evolving technical infrastructure necessary for creating effective GUIs. We explored this topic from three angles, which also intersected with other topic areas such as ML/AI in Visualization Software and Common Platforms:

- **Prioritizing Accessibility.** Current visualization software lacks comprehensive support for diverse physical and cognitive capabilities. Addressing these limitations, as mandated by policies like DOD Section 508, not only meets regulatory requirements but also enhances our software infrastructure. Creating abstract representations in dashboards can significantly aid blind/low-vision users, providing not only improved navigation but also generating rich descriptions for intelligent agents. These abstractions can also contribute to enhancing data and graphical literacy for users without vision issues, presenting promising opportunities for further development. We explored parallels with the classic HCI 'curb cut effect,' contemplating its extension to the realm of visualization software development.

- **Leveraging Generative AI.** The present capabilities of LLM-based generative AI agents offer opportunities to dynamically adapt user interface elements in real-time to user needs. This introduces considerations for multi-modal interactions, such as combining traditional WIMP with voice and/or text. However, the potential of GenAI is constrained by the complexity of user data and workflows. Developing user interfaces for bespoke data and intricate workflows, as seen in scientific research, remains challenging despite the capabilities of GenAI models.

- **Modular, Composable, Adaptable Software.** Rapid iteration of software design and capabilities requires thoughtful architectural considerations. Moving away from monolithic applications to modular components allows for effective representation of GUI elements, facilitating rapid testing of designs, addressing new needs, and building on prior research techniques. While the initial investment in a composable architecture may be significant, it accelerates iteration and prototyping in the long run. This theme is aligned with the common platform topic.

An overarching theme in our discussion revolved around the challenges of conducting high-quality user studies. These challenges stem from participant availability and the necessary investments, both in terms of finances and time. Addressing these challenges head-on presents opportunities to enhance user experiences with visualization tools by leveraging emerging technologies and adopting modern software development practices. By embracing these challenges, we can advance the field and harness the potential of emergent technologies for more effective and user-centric visualization tools.

Topic: Building Communities & Sustainability

Participants: Dominik Moritz, Christoph Garth, Christina Gillman, Alexander Lex, Alex Bock, Marc Baaden, Kenneth Moreland, Katja Bühler

The ultimate goal of most visualization software is that it is useful to a large set of users. Hence, success of a project commonly depends on its adoption and the community that forms around it. In this breakout discussion, we elaborated how we can build sustainable communities around visualization software.

We analyzed a set of successful communities which participants of our group helped co-found, such as the OpenSpace community[7] the Vega community[8] or the TTK community[9].

We identified the following best practices:

- **Rather than building a community, try to join a community.** For example, if there is an existing community related to a new piece of software, it might be advantageous to leverage that existing community and cater to their needs as opposed to attempting to start to build one from scratch.
- **Minimize the start-up burden.** The most difficult thing for a software project is to encourage new users to adopt it. To reduce that initial burden, developers of the project should take all measures to minimize the effort it takes to get started with the project. For example, good getting started tutorials are essential, as are ways to reduce the need to install excessive software packages and dependencies.
- **Identify and engage with key stakeholders.** As a community develops, activity on discussion forums and issue trackers will increase. While it might be tempting to treat all requests equally, successful projects have found it advantageous to give preferential treatment to key stakeholders, e.g., by responding to requests or bug reports from them in a more timely manner. These key stakeholders have the potential to develop into advocates and contributors for the project, possibly even taking on some of the community support tasks thus providing the ability for the project to easier scale beyond its initial set of users.
- **Develop trust by your users.** For a successful community to develop, it is critical that users can trust the software project. For example, documentation should be accurate and up-to-date. Also, project leadership should develop public plans for the future and collect and listen to community feedback. This environment of openness also includes the communication of community guidelines, contributor guidelines, and others.
- **Provide opportunities for learning** Opportunities for learning about the project are essential for adoption. Primarily, these should be in the form of documentation. Good documentation should follow the Diátaxis approach to documentation authoring[10]. These resources should be persistent and searchable.

[7] https://www.openspaceproject.com/
[8] https://vega.github.io/vega-lite/
[9] https://topology-tool-kit.github.io/
[10] https://diataxis.fr/

Another avenue for learning are tutorials at conferences, such as IEEE VIS and domain-specific conferences. Tutorials have the potential to give participants a holistic overview of the project, easing adoption at a later time while simultaneously exposing a large number of potential users to the project.

- **Provide opportunities for discussions.** The ability to ask and answer questions is critical for the success of a community. Successful projects tend to employ both real-time discussion forums with tools like Slack and Discord, as well as asynchronous and searchable methods such as issue tracking and question and answer forums. An important aspect of these tools is their indexability, which makes them easier to find for new users using web search engines.

- **Provide opportunities for participation.** Finally, community members might also want to contribute to core aspects of the project. One way to encourage such participation are Hackatons, where users can contribute to a shared task, moderated by the project leadership.

Other issues discussed include governance models for larger and more mature projects. An example is the governance model for Vega[11], OpenSpace[12], as well as models for funding community building, for example donation models[13].

As an outcome, we discussed a position paper or a blog post, but decided to ultimately organize a panel at IEEE VIS on the topic.

[11] https://github.com/vega/.github
[12] https://github.com/OpenSpace/OpenSpace/.github
[13] https://godotengine.org/donate/

Topic: Typology

Participants: James Ahrens, Marc Baaden, Katja Bühler, Christina Gillmann, Michael Krone, Kresimir Matkovic, Guido Reina, Gerik Scheuermann

We based our discussions on the preliminary results of the preceding Shonan Seminar #145 and the respective publication [22]. One critical observation was that as a community, we take care of periodically surveying our state-of-the-art techniques, often accompanying these publications with an interactive web presence that allows users to select appropriate solutions based on a few properties of the problem at hand. These catalogs, however, do not include the systems and libraries we have developed, and especially do not consider any technical implications or requirements that might be ensue when basing a project about one or even several of these. The scope and utility of such an endeavor is two-fold: on the one hand, we gain a better understanding of the properties and requirements of our software, which also allows relating specific solutions to one another. On the other hand, a concrete implementation will both serve as a portal into our community for an audience outside our domain, and a practical tool for ourselves.

We started defining a hierarchical typology of the most important aspects that categorize visualization software. There was also an extensive discussion on the technical aspects of making this typology (or ontology) accessible to the community. The accessibility and ease of use will be an important step in establishing this typology as a reference model to categorize and compare visualization software in the future. We finally decided that the best way forward would be a flat list of the most relevant aspects and their instantiations and re-aggregate the typology hierarchically afterwards to remove current ambiguities and redundancies. We set up a working repository to start iterate on this after the seminar.

Topic: Grand Challenges in Visualization and Funding

Participants: Kenneth Moreland, David Laidlaw, Gerik Scheuermann, Alexander Lex, Alex Bock, Ana Crisan, Christina Gillman, Kate Isaacs, Jim Ahrens, Hans Hagen, Christoph Garth, Daniel Wiegreffe

A key challenge for visualization software development is funding. However, because funding is highly heterogeneous in different countries, we did instead focus on developing arguments for funding visualization research and software development by identifying and describing the grand challenges of visualization research for the next two decades, updating Chris Johnson's list from more than 20 years ago [16].

- **Visualization Specification / Authoring** Creating visualizations is still difficult to do. While progress has been made in developing GUIs, tools like Tableau are still difficult to use. A potentially promising avenue are natural language interfaces for visualization specification.
- **AI for Visualization** AI and Visualization are both part of the data analysis pipeline, with great potential for their deep integration through methods such as AI and HI (human intelligence) teaming, leveraging the best of both worlds. Also, there are significant opportunities for combining language and visualization, for example, by explaining the content and context of a visualization in language using LLMs.
- **Visualization for AI** The development of AI methods can significantly benefit from visualization approaches. Visualization is critical in model development and understanding, and in aspects related to accountability of decisions made by AI.
- **Cognitive Foundations of Visualization and Theory** What is a good visualization? Which visual encodings, interaction techniques, and integrated systems lead to strong insights? While the community has made progress in understanding low-level aspects of human perception, much work has to be done to understand interactive systems and complex, combined charts.
- **Accessibility** Visualization is used to communicate essential insights in news media, education, and science. Yet, low-vision and blind users, or those with motor or cognitive disabilities are likely to not have equal access to the information contained in these visualizations. Understanding how we can make this information accessible to a diverse set of users is an essential challenge to ensure equitable access.
- **Data and Knowledge**

  Data is an imperfect representation of phenomena of interest. Most of the time, experts that collect data are aware of its limitations and how to interpret it for their purposes. However, this knowledge is often lost when data is stored and repurposed at a later time or analyzed by another person. In practice, re-use of data is incredibly common, yet this key contextual knowledge is not preserved. As a community we need to improve our process of documenting knowledge about data, either in the form of metadata, but

ideally, also "closer to the data", so that the data can be interpreted with that critical knowledge even after a hand-off to other stakeholders.

- **Scientific Software Development** Sustainability and Maintenance, Stewardship of Software
- **Visualization Literacy & Education**
- **Uncertainty**
- **High-dimensional and spatial data** (single cell, lidar, space)
- **Scalability**
- **Display environments**
- **Integration into common software packages** Another challenge is the integration of visualizations into larger software packages such as R or SciPy. The lack of interactivity is particularly problematic here, as this can often only be achieved using additional components such as Javascript. However, this makes the development of these applications more complex.
- **Funding infrastructure work for research**

  Funding for infrastructure such as software maintenance can be difficult to acquire. Different funding agencies have specific criteria for what they fund, so sometimes you have to do this work pro bono alongside the projects. However, this does not make it possible to establish a sustainable infrastructure. A possible solution to these problems is the continuous promotion of the importance of visualization software for science and society, so that there is a stronger demand for stable visualization software.
- **Increase awareness of the importance of visualization**

Topic: Meeting Users Where they Are

Participants: Guido Reina, Hans Hagen, Fritz Lekschas, Seokhee Hong, Kate Isaacs, Steve Legensky, Michael Krone, Gunther Weber, Gerik Scheuermann, Kwan-Liu Ma, Ana Crisan, Takayuki Itoh, Daniel Wiegreffe, Alex Lex, Kresimir Matkovic, Michael Keckeisen

This breakout session featured discussions about meeting the specific needs of both industry professionals and researchers in a diverse range of application domains and how to address these needs as the visualization community.

A key theme was the importance of the immersion of visualization researchers within user communities. One example was using formalized programs such as the Innovative Training Networks (ITN) offered by the European Union. This program would allow embedding researchers, particularly PhD students, in these communities for periods ranging from a week to a year. These "Fellowship" programs have already been investigated and have been found useful. A paper by Hall et al. [11] describes benefits of immersion in visualization research. These instruments can be used to embed visualization researchers within the specific target domain to gather in-depth knowledge and thus foster information exchange. VIZBI and Co-located conferences like BioVis and VizSec were highlighted as positive examples of cross-domain information exchange.

Participants discussed the challenge of integrating visualization tools into existing software ecosystems. The discussion touched upon the use of common tools like Tableau and software languages like Python and R, alongside more bespoke solutions commonly employed in the visualization community. However, these incur significant wind-up costs for creating these demonstrators whereas more common software packages, like ParaView, have their own design restrictions. The widespread use of Python in tooling ecosystems and the ongoing challenges with interactivity were noted. The discussion ended with considerations for stand-alone web development and the proposal of panels at major conferences to continue the dialogue on these topics.

Determining where to publish software-based findings was a pivotal topic. The major opportunities are established venues such as IEEE VIS and the newly created Journal of Visualization and Interaction (JoVI), or alternatively publishing at domain-specific journals were suggested strategies. The idea of "Sustainability badges" in the visualization community was also brought up, requiring further discussion and action. Another idea was to count citations in application science community publications and consider awards for adoption of work outside the visualization community, similar to the existing test-of-time awards.

This breakout session "Meeting Users where They are" provided a fruitful overview of the ongoing challenges and strategies in developing visualization software for diverse user communities. It highlighted the need for immersion within user communities, understanding their software ecosystems, selecting appropriate publication venues, and differentiating between scientific and engineering approaches. The conversation set the stage for ongoing efforts to enhance software development in visualization.

Topic: The Role of AI in Visualization Software

Participants: James Ahrens, Katja Bühler, Anamaria Crisan, Seok-Hee Hong, Takayuki Itoh, David Laidlaw, Kwan-Liu Ma, Kresimir Matkovic, Michael Keckeisen, Claudio Silva, Gunther Weber, Daniel Wiegreffe, Fritz Lekschas, Ivan Viola

The introduction of neural network based artificial intelligence (in the following denoted simply as AI) is disruptive in every technological aspect where digital data play a significant role. Visualization research and its software outcome processes digital data into visual forms that are intuitive to comprehend by human users. Visualization establishes a visual dialog interface between the data and the user with the goal to aid particular use case tasks. Today, AI has already gained significant transformative influence on visualization research and visualization software development.

Within two sessions we discussed the state of the art, best practices and future opportunities to integrate recent developments of AI models into both, visualization applications and their development processes to more effective, human-centered visualization and visual analytics pipelines, but also accelerate and redefine the development process of visualization software itself.

I. AI as an Element of Visualization Applications

Machine Learning for Data Interpretation The Visualization Pipeline is a chain of steps where the data is gradually transformed into visual representations. Along this process, machine learning models assist or perform a fully automated task they are trained for. In the stage of data filtering, data can be denoised using neural-network inference or certain interesting low-level features can be extracted. Such representations might be already suitable for display. In another case, the inference can lead to high-level interpretation into domain-specific objects through semantic data enhancement e.g. by labeling, segmentation, feature extraction and data interpretation that is fed back into the visualization pipeline for display. Visualization can convey these interpretations or serve as an interface where the domain specialist confirms or rejects the presence of specific features, labeled through inference, within the visual data analysis. Especially generative AI methods have also the potential to support reverse engineering of data or models given a specific visualization or parametric model ouput.

AI as an Intelligent Assistant. With the introduction of powerful large language models, an opportunity arises to utilize such models for various assistive functions within visualization software:

- *Conversational Visualization* allows a user to formulate natural-language queries about the data and the large-language model interprets these queries into specific commands within a visualization software. For example, loosely formulated geometric transformations can be instantiated into specific actions by way of fine-tuning the model for completion of a specific task. Furthermore, large-language models can be directly prompted for specific domain meaning or detailed explanation of the visualized structure. In this new workflow, the data analyst can converse with a chatbot to explore and draw conclusions from tabular data, or even medical diagnosis can be established by conversation between the clinician and the chatbot that explains why a certain structure has been characterized as a particular pathology.

- *Automatically generated (textual) help functions.* When fine-tuning the help system of a software with a natural language - formal terminology pairs, large language models can serve as an effective help system that guides the user to system's functionality. For example, ParaView, a well-established scientific visualization software is known for a steep learning curve due to is broad spectrum of flexible functionality. Using ParaView's documentation as fine-tuning, a chatbot can be assistive in operating ParaView through natural language prompts. It can be even used for training naive users who may start their training by asking: "What can I do with this tool?". However, all these functionalities need to be used in caution. The implicit neural network knowledge representations might not lead to correct responses and there is always a non-zero probability that the chatbot response is simply wrong or incomplete. Therefore, utilization of these assistive technologies should be used only in non-critical scenarios, or every outcome has to be scrutinized by respective domain experts who are ultimately responsible for the completion of specific underlying task.

- Modern multi-modal model architectures that can learn from very diverse data in a combined way. This might include different time dependent and potentially spatial data types including user interaction with complex data, user attention and emotion, data, tasks, app configuration etc. This opens new opportunities of research on *user and task adaptive interfaces* including improved interfaces for people with special needs realizing *inclusive* interactive interfaces.

- Supporting and acceleration of visual computing workflows can be achieved by learning and predicting interactions. A specific challenge in this context is the diverse and hierarchical nature of interactions and the related data modelling.

- In the human-computer interaction research community, the term *Human-AI Teaming* relates to handling imperfect machine learning outcomes. This concept of teaming mixes together machine learning for visualization and visualization for machine learning.

- The lack of participants and time is challenging the systematic evaluation of usability of interactive visual applications with or without Human-AI-Teaming. The use of AI to mimic real user behavior, for example by using prediction models to estimate human perception or preference, is an active field of research.

Deep Learning as Methodological Approach to Visualization The introduction of deep learning as a methodology to replace e.g. traditional rendering methods or parts of the rendering pipeline caused a rethinking of traditional approaches computer graphics and visualization research and a wave of novel solutions. One such example is *Differential Rendering*: Neural network training process can be characterized as an optimization process where the numerical chain-rule-based differentiation defines the gradient that is used for optimizer to update the network parameter values. Instead of using implicit neural representations, differentiable approaches can be used on reverting a well-defined algorithmic process that is formulated to be continuous, such that either above

mentioned auto-differentiation can be utilized or analytical gradients can be computed instead. Visualization can be viewed as such a forward continuous process and differentiable methods can be used for estimating the visualization-process parameters. This way, for example, transfer functions can be estimated from a given set of images and data. In principle, any information can be estimated from this optimization process, including the particular steps of the visualization pipeline itself or even the underlying data - along the principle of the neural radiance fields, for example. As visual mapping can be formulated as a specific visualization language grammar, it can be possible to revert-engineer the visualization language, e.g., Vega-Lite, from rendered visualization images.

II. AI assisting the visualization software development process. AI models supporting coding like Github Copilot [14] are already widely used and adopted by developers, though its specific benefits for the development of specialized visualization software is not so clear. Visualization languages and grammars of graphics that emerged in the past, adhere to a specific syntax. As such, this code can be generated by large language models, similar to generation of any other code syntax. But what are potential promising use cases of AI for visualization software development also beyond the current state of the art? During our discussion we identified a couple of promising new use cases and potential directions novel applications of AI to assist Vis software development:

- Increasing interoperability of visualization software by employing AI (LLMs) for transferring/translating code, between different third party dependencies
- Using AI for to automated code / library / dependency reduction
- Employing AI to "reverse engineer" visualization code e.g. from publications (published without code):
  - From pseudo code to code: Presenting the AI pseudo code from a paper to recreate the code for the visualization in a specified programming language.
  - From visualization to code: Presenting the AI a figure with a specific chart to automatically generate respective code to recreate that chart for given data.
  - From paper to code: Presenting the AI a Vis paper to recreate the underlying application.

III. Practical challenges and consideration for the integration of AI into visualization software The group identified several points to be considered when integrating AI into visualization software.

- Tasks where AI can help with have to be carefully chosen
- A measurable positive impact, as well as correctness and reliability of the AI components should be ensured.
- Thinking beyond LLMs might be beneficial including e.g. recent work on multi modal foundation models

[14] https://github.com/features/copilot

- The choice of AI foundation models to be integrated in visualization software is non-trivial. Considerations include many dimensions including specialization, accessibility and openness of the models as well as data privacy.
- Architectural consideration in using AI and Vis in parallel to ensure efficiency
- Interfacing to industry standards for data types is favorable

Concerns in respect to the limits of the usage of AI models were related to the still very fast evolving field of AI and that the sustainability of solutions is still not fully given. Also the use of AI models in connection with multi modal interfaces requires care in respect to ensure determinism, reliability and interpretability/causality of outcomes.

Topic: How can visualization software be FAIR?

Participants: Christoph Garth, Jim Ahrens, Alex Bock, Fritz Lekschas, Marc Baaden, Gerik Scheuermann, Claudio Silva

FAIR requirements have strongly increased in funding calls and publications outlets. The group identifies two major implications for visualization software: First, as visualization researchers produce software as a primary class of research artifacts, the question naturally arises how VIS software can be published in a FAIR manner and which requirements one should fulfill to achieve this goal. A second question is, how can VIS software help domain scientists outside the visualization field in producing and maintaining FAIR data. Due to scope, the group decides to focus on the first issue.

Considering the individual facets of the FAIR principles, the group brainstorms how these can be translated into practical recommendations for VIS researchers to achieve a gradually increasing degree of FAIRness in software publishing and dissemination. These recommendations will form the basis for an article, targeted at the Visualization Viewpoints category in Computer Graphics & Applications, and an implementable pre-publication checklist for researchers.

Topic: What, How, and Where to Publish

Participants: Alex Bock, Ana Crisan, Cristophe Garth, Kate Isaacs, Michael Krone, David Laidlaw, Fritz Lekschas, Ken Moreland, Dominik Moritz, Ivan Viola, Gunther Weber, Daniel Wiegreffe

We have recognized software tools as an essential part of visualization research, and yet there are several barriers to sharing, reusing, and maintaining them. We discussed several ways to encourage the sharing, reuse, and maintenance of visualization through forms of publication. These included changes to publication processes and expectations and additional venues in which software contributions could be recognized. With each of these pointes, we further discussed what should be published and how they should be published—what mechanisms and requirements should each form entail. Finally, we set forth a plan for realizing the outcomes of our discussion.

Software-focused publications are well established in other communities like bioinformatics and data science. For instance, the Oxford Bioinformatics journal has a publishing format called Application Notes. An application note paper entirely focuses on the software tool and comprises only two pages. Such papers do not present any new scientific results but instead typically provide a short use case that demonstrates the utility of the software tool. Similarly, Journal of Open Source Software is a journal that focuses entirely on software-focused papers. Given its focus, the journal implemented software-specific requirements such as code archival, tests, or community guidelines. Additionally, authors have to write a one to two page description of their tool, how it's filling a gap, and how it relates to existing software. These existing venues allow us to learn how a successful software-focused publication tier can be established. But while visualization software publication would have much in common with publications at the above mentioned venues, we recognized that visualization software has special requirements (e.g., viewing device or user study) which warrants a new format tailored to the visualization community.

We noted there are several scenarios of visualization software, from general libraries to be used by the visualization community and beyond to more focused visualization software aimed at specific communities. These different types of software and users suggest different types of validation are needed. Rather than ranking the level of reusability or availability, we agreed that different types of reusability and availability could be considered to fit these different contexts.

A live demonstration session at visualization conferences would highlight the potential reuse of software. In the near term, a workshop or application spotlight-type event may provide the right balance of incentive and additional work by the authors. Hands-on and bring-your-own-data demonstrations are something we want to encourage in the longer term.

Archived publications in well-known venues represent strong incentive and recognition for the work that reusable software entails. In the medium term, there is strong support for an additional type of short paper that focuses on software contributions. Leveraging and adapting the mechanisms and requirements of Bioinformatics Application Notes and the Journal of Open Source Software as described above is a good starting point. We note these short papers should not preclude papers about the scientific contributions implemented by the software, but instead recognize the contributions of the software itself.

Additionally, we discussed whether traditional full-length papers should require software artifacts such as source code, executables, or runnable containers. It was noted that IEEE Transactions on Visualization and Computer Graphics already has the option of a Replicability Stamp, where software is reviewed after a paper is accepted for publication. This process has revealed the breadth of needs in verifying research software.

Topic: Common Platform, Building Blocks / Modularization and Interop / Integration; connecting vis software / systems / building blocks; APIs

Participants: Marc Baaden, Steve Legenski, Kenneth Moreland, Michael Krone, Guido Reina, Katja Bühler, Ivan Viola, Claudio Silva, Katherine Issacs, Seok-Hee Hong, Dominik Moritz, Fritz Lekschas, Daniel Wiegreffe, Kwan-Liu Ma, Anamaria Crisan

While we had initially split the topic into the three more specific subtopics, in the plenary discussion we concluded that the distinction would hinder the discussion and therefore merged them into a single one to be discussed in multiple iterations.

During the first iteration of the discussion, we discussed the problem from a workflow perspective, citing Alteryx or Galaxy and CWL (among many other open approaches) and the respectively required building blocks, leading to importance of proper interfaces and APIs as a valuable contribution. The lessons learned from practice suggest that the advent of Apache Arrow and its incarnations and formats like Pandas and Parquet has been a game changer for its users, allowing for high-performance, zero-copy usage and slicing of massive data sets. The ensuing question is what such an endeavor would look like for the scientific visualization community, since tables are not necessarily an appropriate abstraction, and additional features like streaming and hierarchies or LOD levels are required. The intermediate take-away was that we should invest more time into designing the APIs and interfaces of our software artifacts, since these efforts both pay off significantly in practice and represent core take-away messages that are valuable in publications (see also page 29).

In the subsequent discussions, the audience was slightly biased towards classical scientific visualization and thus both had a different state of the art and different views of the subject. The discussion quickly concluded that a single, common platform is an unattainable panacea. Although standard tools are established for specific domains, even there the need for customized tools exists.

The discussion therefore focused more on the general needs of platform building for visualization tools. What should be the general structure of a common platform? Should it be one monolithic tool? Such large repositories allow for well integrated components, efficient interoperability, and simplified dependencies. But such repositories also have major development burdens and require users to adopt large, unruly libraries or applications for even small tasks. At the other end of the spectrum is to break everything into small, independent components. Although more agile, it is usually difficult for such tools to interact with each other, and doing so may require wasteful transformations (cf. the discussion about Arrow above). Furthermore, dependency management is an issue, and the use of many tools increases the risk that any one of them will become unavailable. However, this strategy represents a convenient stepping stone to meeting the user in their own ecosystem, a topic that was also discussed at this meeting (see page 23). In short, there is no free lunch, and it is an engineering trade-off to determine where to draw component boundaries.

The participants' experience suggested that attempting to design a specific solution and foist it upon the community is folly. Instead, the discussion focused

on broad issues and solutions while pulling in experiences of tools and techniques that work best. Rather than try to solve what the "right" level of building blocks should be, the group recognized that some level of component division is assured and focused on how such visualization tools might interact. The discussion then focused on two main discussion points: a common data model and a common interaction layer.

Common Data Model Any interesting interaction between visualization software components inevitably requires sharing data. Doing so is only possible if components understand each other's data. Although the base representation of data can usually be trivially broken into tables or arrays, such low level structures provide little information about the semantics required to interpret them.

For inspiration for connecting components together, the group looked at the in situ visualization community [3]. The primary challenge of in situ visualization is the coupling of a scientific solver code with visualization code, which are independently constructed and often have conflicting data models. Much thought as been given on what this interface looks like, particularly with respect to the data model, and several ideas that have worked to varying degrees.

What the participants found that works particularly well with respect to data sharing is an approach that uses a simple library called Conduit [12]. Conduit provides an object description using a hierarchy of key/value pairs, where the values may reference an array. The structure is similar to that in many "self-describing" data stores such as JSON, YAML, HDF5, ADIOS, and others. However, Conduit is separate from any other data storage or functionality; it provides a data object that can be passed among software components, and that is it. Conduit has been adopted by several in situ visualization libraries, such as Ascent [18], Catalyst [2], and Kombyne [19], to simplify the passing of data from simulation to visualization libraries.

The group suggests a similar approach of adopting a lightweight description object to annotate data as a flexible mechanism for joining disparate components. Of course, an alternate tool to Conduit could be designed, but the existing functionality seems well suited for more general application.

Although this approach of adding attributes to arrays is referred to as self-describing, this structure alone is not sufficient for software to make use of such data. To make sense of such data, the software components must agree on a schema for the structure. This provides a specific convention for names or other attributes that consuming components can use to adapt the data to their own data models. In situ visualization tools rely on a schema called Blueprint [12], which provides mechanisms to describe common data structures of the scientific visualization community. Other visualization communities might require alternate or expanded schemas. A suggested approach is to observe the data models of mature tools and use this as a template for the initial structure.

Common Interaction Model Given multiple visualization components that share some data, looking into user interaction is the next logical step. Ideally, states like selection or filtering should be shared or transmitted between components via some mechanism. Again, this hinges on the definition of interfaces or APIs that share a common formulation of interactions and intents and take

of the technical details. An orthogonal approach to the above would first need to establish the visualization parameters that can sensibly transported from one component to another. Then it should design an implementation that, for example, allows to robustly ignore interactions that are irrelevant or not yet implemented and take part in the rest of the communication. This effectively describes a generalization of the implementation in Heinemann et al. [14]. A first internal draft of this is outlined by G. Reina. The connection layer could be used both for rapid prototyping of coupled visualizations that do not coexist in the same framework or to facilitate reproducibility. The concept is based on synchronized arrays of arbitrary objects that can be named to enable mapping across different data and tools. This would allow to synchronize entities without requiring access to data beyond that is interesting for each implementation. For example, one of several isosurfaces could be selected in ParaView, resulting in its respective ID. This ID could be used to select marks (lines or bars etc.) in a vega-lite diagram that depicts metrics or other metadata that is semantically connected to the isosurface. Both tools only need to load and manage data pertinent to the respective visualization primitives and still contribute to a sensible meta system for the analysis of the given dataset. So far, the more obvious state to share would comprise camera pose and parameters, light source pose and parameters, transfer functions (both in parametric form and "baked" into textures), and flags that represent selection and filtering state. The draft envisions a broker component that mediates between the different running components and would also allow the user to explicitly perform the mapping between the known entities in all involved components (i.e., one component might call it 'Camera1' while the other uses 'PerspectiveView' etc.). The obvious drawback of such a solution is that the user is ultimately responsible for connecting entities the IDs of which are consistent and semantically relevant. It would be interesting to investigate technical and semantic descriptors that allow for a modicum of validation to simplify usage.

Topic: Education of PhD students for visualization software development

Participants: Michael Keckeisen, Krešimir Matković, and Dominik Moritz

Our discussion revolved around the education of graduate students, specifically those pursuing PhDs with a research focus. We emphasize that our focus is not on undergraduate or master's level students since they lack a research orientation.

We advocate for a comprehensive approach to graduate education in visualization software development, integrating research principles, engineering practices, and collaborative learning experiences. By equipping students with these skills, we aim to foster a new generation of researchers capable of making meaningful contributions to the field.

More specifically:

The work should be research. Target contributions that generate new human knowledge. Students must develop software capable of demonstrating research insights, supporting instrumentation, and facilitating adaptability. Depending on the goals/research type the software can be scrappy or should be built well. The skills also differ based on the contribution. Students also come in with different backgrounds. Their skills/excitement should be leveraged.

Follow good engineering practice. They lead to better research and save time in the end. When starting a project, think about what practices are the right ones for the research at hand. For example, always do version control. If needed, have continuous integration, tests (UI and unit tests), build documentation, processes for contributors/code reviews, pull request templates etc. Building software alone is very different from building with two, or more people. It's a good practice from a pedagogical perspective to have (at least) two students work together. One danger is that sometimes there is a star student where others just get out of the way. But students need to learn to work together.

Critical thinking, analysis, and debugging are hard to teach. They are essential, and should be prioritized.

Students benefit from working together. A challenge when students work together is when one is not able to articulate their contribution. Then it's difficult for them to establish themselves as independent researchers. To overcome this issue, each student needs to carve out their focus. However, at the same time if you have students who all contribute, each of them can carve out their own focus. A huge benefit is that credit does not split 50/50 but instead 70/70. The pie grows.

Some different kinds of research that involve software.

1. Code for running a study.
2. Software for an algorithm that needs to be benchmarked. Analysis scripts. Reproducibility is useful. Contribution: new algorithms, user studies.

3. Doing tool driven research where a holistic software artifact is the contribution. Reuse of the software is an explicit goal. Contribution: new workflow that makes a qualitative or quantitative improvement
   (a) Software that a collaborator used to do some task. Ideas live on but the code does not.
   (b) Software that is sustainably built.

Some ways in which students can learn engineering practices. Classes, learn from their advisor who enforces certain practices, collaboration projects with companies that require certain qualities from the artifacts that are being produced, learn by example from an advisor (copy your mentors), do a research or class project that involves engineering practices where students can learn, and internships (e.g. internships at a place with strong engineering practices).

Topic: What are promising technologies to look out for / to adopt right now?

Participants: Ivan Viola, Kresimir Matković, Christoph Garth, Marc Baaden, Michael Keckeisen

The discussion centered around identifying promising technologies in the realm of visualization that should investigated for adoption, particularly focusing on the appropriate abstraction level for developing visualization tools and platforms. This summary organizes the insights from the discussion into multiple themes.

Visualization Languages and Abstraction Levels. The group underscores the importance of finding the right abstraction level for creating effective visualization tools. There's a consensus on the need for a "Vega-Lite for 3D," indicating a desire for tools that simplify the creation of complex 3D visualizations without sacrificing power or flexibility. It is identified that there is a gap in tools specifically designed for 3D visualization that are as intuitive and powerful as Vega-Lite is for 2D.

Rendering Technologies. A significant portion of the discussion is dedicated to rendering technologies, with web-based graphics and WebGPU taking center stage. WebGPU is lauded for its support of all shader types, compute shaders for physics and inference, and its portability across platforms, including Android Chrome. This technology, especially when combined with WebAssembly (WASM), is seen as a promising avenue for "write once, run everywhere" visualization applications while simplifying their development. The success story of Nanographics, which utilized an abstraction layer for portability across OpenGL, Vulkan, and WebGPU, is highlighted as an example of effective use of these technologies. There is curiosity about the integration of WebGPU with game engines like Babylon.js and potential shifts from other engines like three.js to embrace WebGPU.

Game Engines and other Middleware. Game engines such as Unreal are recognized for their standard features and ease of handling 3D graphics and scene graphs, which are cumbersome to develop from scratch. The discussion also touches on *Houdini* and *ytini*, particularly their use in rapid prototyping and astrophysics visualization, respectively, showcasing the versatility of game engines and specialized software in creating high-quality visualizations. The Universal Scene Description (USD) file format is mentioned as a significant innovation for scene assembly, alongside the ANARI cross-platform 3D rendering engine API, which is anticipated to be the foundation for future tools like Paraview. This suggests a movement towards standardization and cross-compatibility in 3D visualization tools.

Hardware Advancements and Other Notable Technologies. Apple's Vision Pro is singled out as a game-changer in hardware, suggesting that new hardware capabilities could significantly influence the development and deployment of visualization technologies. Additional software technologies discussed include Kokkos for C++ performance portability, targeting various platforms

from CUDA to SYCL, and NeRFs (Neural Radiance Fields) for creating life-like 3D models from photographic images. These technologies represent the cutting-edge in performance optimization and realistic rendering, respectively.

Topic: AR/VR & Unconventional Displays

Participants: Dominik Moritz, Alexander Bock, Ivan Viola, Gunther Weber, Takayuki Itoh, Alexander Lex, Daniel Wiegreffe

The group began by discussing the new Apple Vision Pro VR/AR headset. That led to a list of uses for VR in the context of visualization and some speculation on challenges for those uses.

Potential uses include:

- Visual exploration of scientific data, especially that with intrinsic 3D geometric relationships, multiple values at each spatial location, variation in time, and scale beyond what can be viewed on a desktop display. Examples include 3D biological or medical imaging, large simulation results, planetary-scale remote sensing
- Visualization applications that would benefit from very large 2D displays.
- Some 2D visualization approaches, including overdrawing and binning, may benefit from judicious and understated use of 3D and stereo
- Collaborative visualization both remote and co-located.
- Teleoperation.
- Augmented reality for situational awareness in the real world and in visualization applications.
- A super nice office anywhere you go: home, workplace, cafe, park, hotel, plane.

There are a number of challenges that will influence the success of these potential uses.

Development infrastructure. Currently, most VR visualization applications are custom-developed for a small group of collaborators. Development can take many months or years, and that limits the number of collaborators and the impact that VR/AR visualization can have. More infrastructure to speed up software development would accelerate progress.

High maintenance costs. Traditionally, high-pixel-count VR visualization has required a cave or a finicky and expensive headset. Both scenarios involve substantial maintenance costs. Apple's new headset may achieve sufficient resolution in a relatively low-cost package to allow high-pixel-count VR in a low-maintenance setting.

Travel. In addition to the startup costs of software development, many VR visualization applications can only run in specific locations. Traveling to a display location, even within a single building, has been a barrier to regular use.

User input. VR applications typically make common input devices, like a keyboard, unusable. Replacement input mechanisms are usually less efficient, making regular use less appealing. Spoken input, virtual tablet interfaces, and other UI techniques may help but have not yet solved this problem.

Collaboration. Synchronizing the virtual world across users so that they can meaningfully communicate about it is another challenge.

Productive integration. A VR session can be a "wow" experience, but it is often difficult to take away more than the "wow" memory. Better integration into scientific and professional workflows remains a challenge.

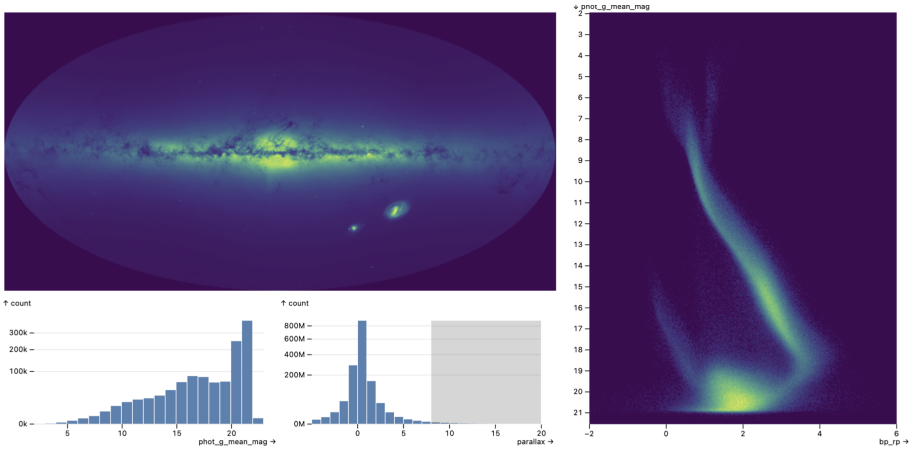

Figure 1: A Mosaic-based interface for interactive visual exploration of all 1.8 billion stars in the Gaia star catalog. A high-resolution density map of the sky reveals our Milky Way and satellite galaxies. Stars with higher parallax values are interactively selected, forming a Hertzsprung-Russell diagram of color versus stellar magnitude on the right. Mosaic offloads density and histogram computation to a backing scalable database, and automatically builds optimized data cube indexes to support interactive linked views.

Visualization Software Showcases

Participants: all participants
Presenters: Alexander Bock, Fritz Lekschas, Steve Legensky, Alexander Lex, Dominik Moritz, Ivan Viola

In this plenary session, the participants were given the opportunity to showcase some examples of successful or upcoming visualization software, either by giving a short presentation or by letting the other participants try out their tool. The first presenter was Alexander Bock who showed *OpenSpace* (OpenSpace, [5]), followed by Dominik Moritz presenting *Mosaic* (see Figure 1). Fritz Lekschas showed *AnyWidget* (Figure 3, [21]) and Alexander Lex gave a preview of an upcoming tool that is not yet published. Steve Legensky showed an example of a web-based visualization application for monitoring and steering HPC systems. Ivan Viola gave an overview of a number of web-based molecular visualization and modeling applications (WebGPU-based Systems for Modeling and Visualization of Cellular Mesoscale) that either resulted from student works or research projects and gave the other participants the opportunity to try out an intelligent, conversation-based user interface. Below are short descriptions of some of the tools that were presented.

Mosaic [13] is an architecture for greater scalability, extensibility, and interoperability of interactive data views. It decouples data processing from specification logic: clients publish their data needs as declarative queries that are then managed and automatically optimized by a coordinator that proxies access to a scalable database. Mosaic makes order-of-magnitude performance improvements

over existing web-based visualization systems—enabling flexible, real-time visual exploration of billion+ record datasets. Figure 1 shows an example application built with Mosaic. You can try this example (albeit on a sample of the data) in a web browser at uwdata.github.io/mosaic/examples/gaia.html. Mosaic clients can easily be implemented for existing or new visualizations. Mosaic has the potential to be an open platform that bridges visualization languages, scalable visualization, and interactive data systems more broadly.

OpenSpace OpenSpace [5, 4] is a long-standing software package developed by Linköping University, the American Museum of Natural History, the University of Utah, and New York University in collaboration with a large number of digital science centers all over the world. The software is designed to serve as a tool for visualization and astronomy research, while simultaneously functioning as a software used for the dissemination of research findings to the general public, while also spuring interest in STEM fields.

In many ways it can be compared to a custom game engine in which many modalities of datasets (geolocated raster or vector datasets, spatial pointcloud data, volumetric simulations, 3D models, and more) can be rendered in a single coordinate system and thus be placed within their proper context. This common context allows for seamless transitions from sub-meter resolution on planetary surfaces all the way to the edge of the universe at the cosmic microwave background radiation.

The existing plugin infrastructure enables the rapid development of new features into the system and is flexible enough to support the integration of third-party rendering tools, such as yt, ViaMD, and others. Using these third-party softwares prevents the need to reinvent the wheel for every rendering technique while simultaneously providing access to immersive display environments to these tools which traditionally lack these capabilities.

AnyWidget AnyWidget [21] is a small library developed for the single purpose of making it easier for developers to create interactive widgets for Jupyter Notebook/Lab and Google Colab. Widgets are a powerful way to embed visualization directly into data science workflows. However, building widgets had been an error-prone and time-intensive process. AnyWidget makes this process drastically more efficient by requiring close to no setup. Additionally, it provides modern tooling such as hot-module reloading know in the web development community.

For instance, as shown in Figure 3, with AnyWidget we can build a widget for Observable Plot's scatter plot requiring only six lines of Python code and twenty lines of JavaScript code.

Persist—Persistent and Reusable Interactions in Computational Notebooks Computational notebooks, such as Jupyter, support rich data visualization. However, even when visualizations in notebooks are interactive, they still are a dead end: Interactive data manipulations, such as selections, applying labels, filters, categorizations, or fixes to column or cell values, could be efficiently applied in interactive visual components, but interactive components typically cannot manipulate Python data structures. Furthermore, actions performed in

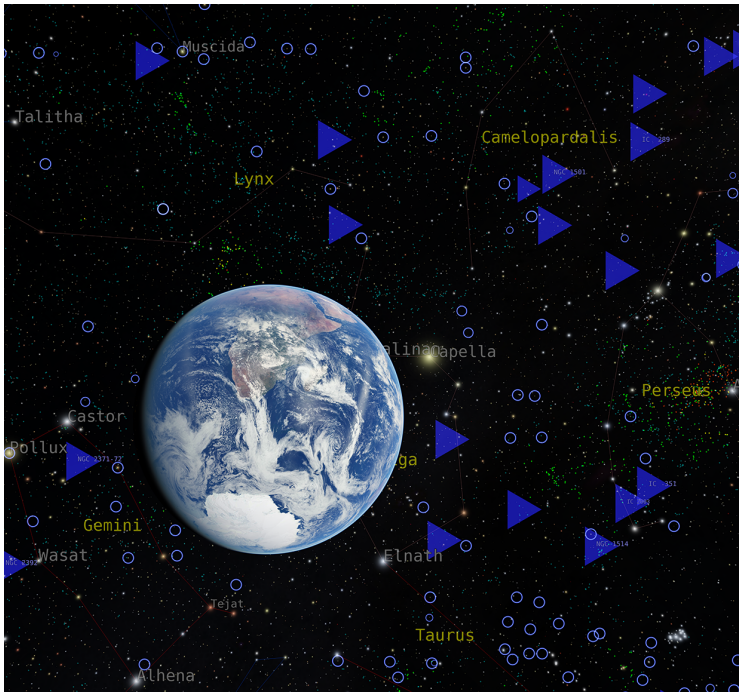

Figure 2: A rendering from OpenSpace showing the Earth bathed in its surrounding data, showing the location of star clusters, exoplanetary systems, and planetary nebulæ.

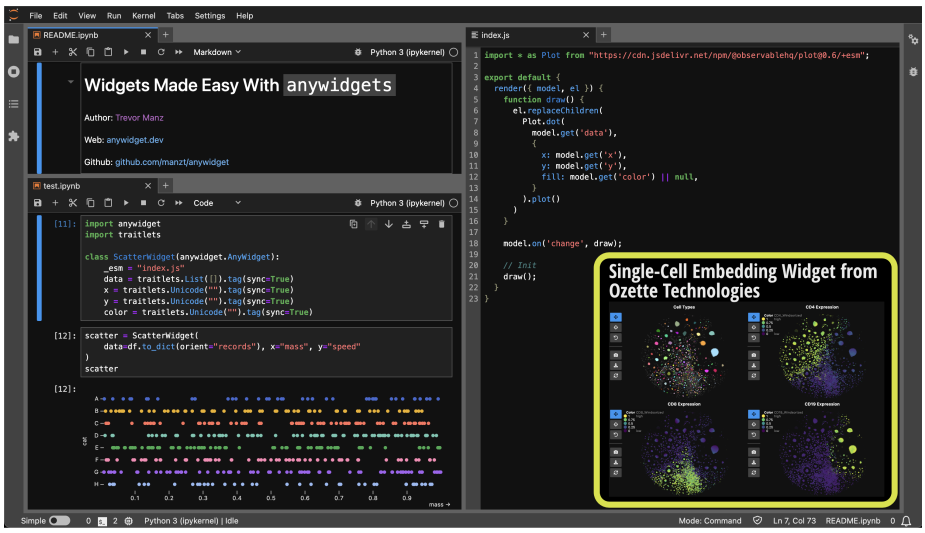

Figure 3: An example widget for integrating Observable Plot's scatter plot with Pandas DataFrame. The popout in the bottom right shows a fully-featured interactive scatter plot widget developed with AnyWidget by Ozette Technologies [9] that shows a single-cell embedding visualization of immune cells, which were clustered with FAUST [8].

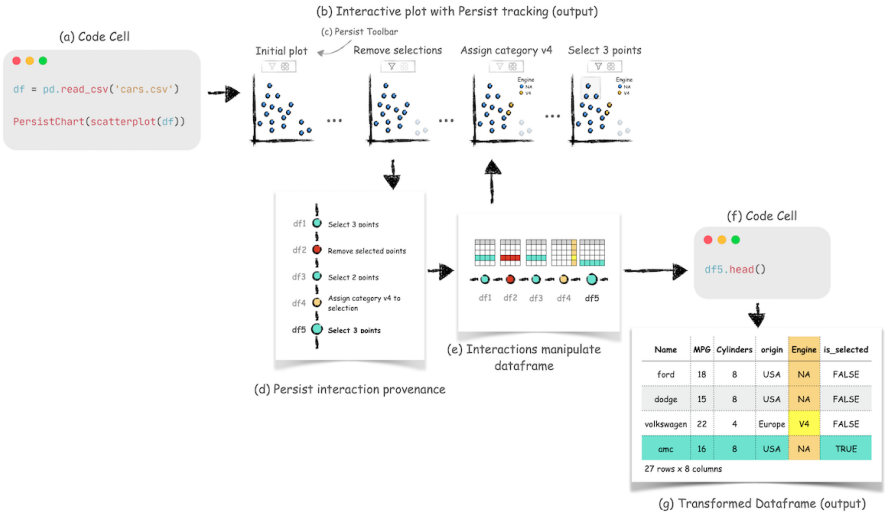

Figure 4: The persist workflow of making interactions in Jupyter persistent.

interactive plots are volatile, i.e., they are lost as soon as the cell is re-run, prohibiting reusability and reproducibility. To remedy this, we introduce Persist [7], a family of techniques to capture and apply interaction provenance to enable persistence of interactions. When interactions manipulate data, we make the transformed data available in dataframes that can be accessed in downstream code cells. We implement our approach as a JupyterLab extension that supports tracking interactions in Vega-Altair plots and in a data table view. Persist can re-execute the interaction provenance when a notebook or a cell is re-executed enabling reproducibility and re-use. The code and documentation is available at https://github.com/visdesignlab/persist/.

WebGPU-based Systems for Modeling and Visualization of Cellular Mesoscale WebGPU can now be officially used in most common browsers, for simple graphics applications but also for advanced applications suitable for a particular science domain. Presented demos have shown applications developed at KAUST which fulfill the task of 3D volume visualization of cryo-electron tomography data, or for procedural modeling of cellular mesoscale models. Figure 5 depicts the complete, procedurally modeled, and scientifically accurate ultrastructure of the SARS-CoV-2 virion. Such ultrastructure can be visually explored by non-expert audience through a conversational visualization interface.

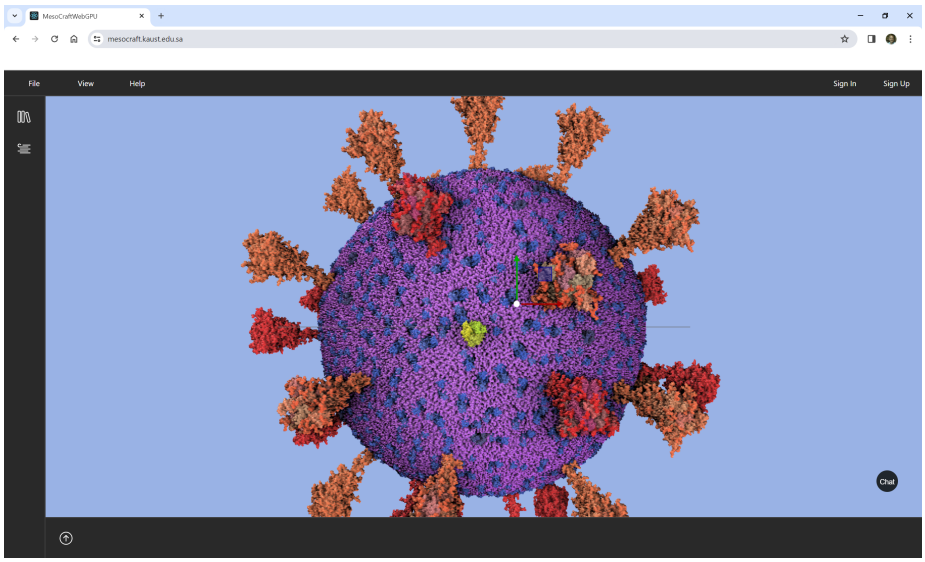

Figure 5: Procedurally modeled SARS-CoV-2 virion in a WebGPU-powered modeling application.

Summary of new findings

In the seminar we made several findings.

The Gap between Visualization and Applications is manifold. In the presented talks it became clear that all participants of the meeting face the gap between visualization and applications. Still, the way they experience this gap is very manifold. It ranges from problems when trying to publish application-oriented visualization approaches in visualization venues, over missing funding for software development tasks to a lack of recognition for ready-to-use visualization approaches. A main contribution of the seminar is a list of problems that have to be solved in the given context.

There exist attempts to close the Gap between visualization and Applications. In the seminar we discussed the current efforts in bridging the existing gap. They exist in terms of encouraging members of our community to provide code of their visualization approaches with their papers or the test of time award at the IEEE Vis conference. The participants agreed that we need take make more use of these offers in terms of bridging the gap between visualization and applications.

There needs to be more efforts in closing the Gap between visualization and Applications. Although there exist attempts to bridge the considered gap, the participants made significant effort in defining and suggesting new ways to bridge the gap between visualization and applications. These include mandatory submission of code, novel tracks in the visualization conferences and further ways of honoring the extra work that comes with applicability.

Identified issues and future directions

In the seminar we identified several issues, that have been brought up in the individual talks, discussed in break out sessions and roughly formalized in this report. The main ongoing task is to formalize the found problems, find potential solutions for them and make an effort in implementing them into the daily work of visualization researchers. In the seminar different subgroups have formed that committed to continue on various sub-problems in the given context, namely:

- A taxonomy of visualization software
- An application paper track
- Structured funding for software development
- Building a community

Therefore, the organizers of the Shonan Seminar aim to continue the work on this important topic by organizing further events such as the VisGap Workshop at EuroVis, joint publications on the topic and further seminar series at Shonan or at similar venues.

References

[1] Y. Annanias, D. Zeckzer, G. Scheuermann, and D. Wiegreffe. An interactive decision support system for land reuse tasks. *IEEE Computer Graphics and Applications*, 42(6):72–83, 2022.

[2] U. Ayachit, A. C. Bauer, B. Boeckel, B. Geveci, K. Moreland, P. O'Leary, and T. Osika. Catalyst revised: Rethinking the paraview in situ analysis and visualization API. In *High Performance Computing*, pages 484–494, June 2021.

[3] A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock, and E. W. Bethel. In situ methods, infrastructures, and applications on high performance computing platforms. *Computer Graphics Forum*, 35(3):577–597, June 2016.

[4] A. Bock, E. Axelsson, K. Bladin, J. Costa, G. Payne, M. Territo, J. Kilby, M. Kuznetsova, C. Emmart, and A. Ynnerman. OpenSpace: An open-source astrovisualization framework. *The Journal of Open Source Software*, 2(15), jul 2017.

[5] A. Bock, E. Axelsson, J. Costa, G. Payne, M. Acinapura, V. Trakinski, C. Emmart, C. Silva, C. Hansen, and A. Ynnerman. OpenSpace: A System for Astrographics. *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[6] S. Eichelbaum, M. Hlawitschka, and G. Scheuermann. Openwalnut: An open-source tool for visualization of medical and bio-signal data. *Biomedical Engineering/Biomedizinische Technik*, 58(SI-1-Track-G):000010151520134183, 2013.

[7] K. Gadhave, Z. Cutler, and A. Lex. Persist: Persistent and reusable interactions in computational notebooks. 2024.

[8] E. Greene, G. Finak, L. A. D'Amico, N. Bhardwaj, C. D. Church, C. Morishima, N. Ramchurren, J. M. Taube, P. T. Nghiem, M. A. Cheever, et al. New interpretable machine-learning method for single-cell data reveals correlates of clinical response to cancer immunotherapy. *Patterns*, 2(12), 2021.

[9] E. Greene, G. Finak, F. Lekschas, M. Smith, L. A. D'Amico, N. Bhardwaj, C. D. Church, C. Morishima, N. Ramchurren, J. M. Taube, P. T. Nghiem, M. A. Cheever, S. P. Fling, and R. Gottardo. Data Transformations for Effective Visualization of Single-Cell Embeddings, July 2022.

[10] S. Grottel, M. Krone, C. Müller, G. Reina, and T. Ertl. MegaMol - A Prototyping Framework for Particle-based Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 21(2):201–214, 2015.

[11] K. W. Hall, A. J. Bradley, U. Hinrichs, S. Huron, J. Wood, C. Collins, and S. Carpendale. Design by immersion: A transdisciplinary approach to problem-driven visualizations. *CoRR*, abs/1908.00559, 2019.

[12] C. Harrison, M. Larsen, B. S. Ryujin, A. Kunen, A. Capps, and J. Privitera. Conduit: A successful strategy for describing and sharing data in situ. In *EEE/ACM International Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*, Nov. 2022.

[13] J. Heer and D. Moritz. Mosaic: An architecture for scalable & interoperable data views. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[14] M. Heinemann, S. Frey, G. Tkachev, A. Straub, F. Sadlo, and T. Ertl. Visual analysis of droplet dynamics in large-scale multiphase spray simulations. *Journal of Visualization*, 24(5):943–961, Oct. 2021.

[15] S. Jänicke, C. Heine, and G. Scheuermann. Geotemco: Comparative visualization of geospatial-temporal data with clutter removal based on dynamic delaunay triangulations. In *Computer Vision, Imaging and Computer Graphics. Theory and Application: 7th International Joint Conference, VISIGRAPP 2012, Rome, Italy, February 24-26, 2012, Revised Selected Papers*, pages 160–175. Springer, 2013.

[16] C. Johnson. Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, 2004.

[17] M. Kampfrath, R. Staritzbichler, G. P. Hernández, A. S. Rose, J. K. Tiemann, G. Scheuermann, D. Wiegreffe, and P. W. Hildebrand. Mdsrv: visual sharing and analysis of molecular dynamics simulations. *Nucleic Acids Research*, 50(W1):W483–W489, 2022.

[18] M. Larsen, E. Brugger, H. Childs, and C. Harrison. Ascent: A Flyweight In Situ Library for Exascale Simulations. In *In Situ Visualization For Computational Science*, pages 255 – 279. Mathematics and Visualization book series from Springer Publishing, Cham, Switzerland, May 2022.

[19] S. M. Legensky, E. P. Duque, D. A. Amels, B. Whitlock, M. Meyer, A. Gerstenberger, P. Adami, and R. Thümmler. Industrial and biomedical CFD workflows enhanced via coprocessing for knowledge capture and computational steering. In *Eleventh International Conference on Computational Fluid Dynamics (ICCFD11)*, July 2022.

[20] K. Ma, I. Liao, J. Frazier, H. Hauser, and H. Kostis. Scientific storytelling using visualization. *IEEE Computer Graphics and Applications*, 32(1):12–19, 2012.

[21] T. Manz. anywidget.

[22] G. Reina, H. Childs, K. Matković, K. Bühler, M. Waldner, D. Pugmire, B. Kozlíková, T. Ropinski, P. Ljung, T. Itoh, E. Gröller, and M. Krone. The moving target of visualization software for an increasingly complex world. *Computers & Graphics*, 87, 2020.

[23] F. Sauer, T. Neuroth, J. Chu, and K. Ma. Audience-targeted design considerations for effective scientific storytelling. *Comput. Sci. Eng.*, 18(6):68–76, 2016.

[24] M. Söchting, M. D. Mahecha, D. Montero, and G. Scheuermann. Lexcube: Interactive visualization of large earth system data cubes. *IEEE Computer Graphics and Applications*, 2023.

[25] D. Vartziotis and J. Wipper. *The GETMe Mesh Smoothing Framework: A Geometric Way to Quality Finite Element Meshes*. CRC Press, 2018.

[26] A. Wiebel, C. Garth, M. Hlawitschka, T. Wischgoll, and G. Scheuermann. Fantom-lessons learned from design, implementation, administration, and use of a visualization system for over 10 years. 2009.

[27] Y. C. Ye, F. Sauer, K. Ma, A. Konduri, and J. Chen. A user-centered design study in scientific visualization targeting domain experts. *IEEE Trans. Vis. Comput. Graph.*, 26(6):2192–2203, 2020.