The Gap between Visualization Research and Visualization Software (VisGap) (2020)
C. Gillmann, M. Krone, G. Reina, T. Wischgoll (Editors)

Framing the challenges of operational and domain usage of volume visualization methods in ocean science

K. Bemis

Department of Marine and Coastal Sciences, Rutgers University, U.S.A.

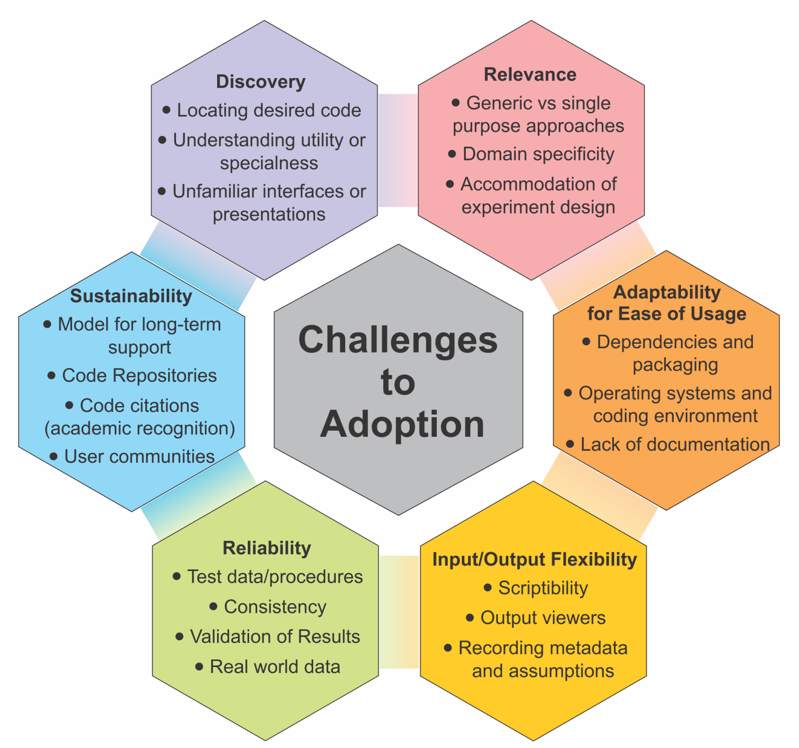

Discovery
• Locating desired code
• Understanding utility or specialness
• Unfamiliar interfaces or presentations
Relevance
• Generic vs single purpose approaches
• Domain specificity
• Accommodation of experiment design
Sustainability
• Model for long-term support
• Code Repositories
• Code citations (academic recognition)
• User communities
Challenges to Adoption
Adaptability for Ease of Usage
• Dependencies and packaging
• Operating systems and coding environment
• Lack of documentation
Reliability
• Test data/procedures
• Consistency
• Validation of Results
• Real world data
Input/Output Flexibility
• Scriptibility
• Output viewers
• Recording metadata and assumptions

Figure 1: *Challenges to the adoption of visualization advances by scientific domain experts in ocean science.*

Abstract

Several case studies are used to explore why the adoption of visualization software, especially for the visualization of 3D time-varying ocean data, has lagged behind the development of visualization techniques. The development history of the Silver and Wang feature tracking for time-varying 3D volume data highlights the challenges of decadal scale development and support. The experiences of supporting operational use of processing and visualization for the COVIS oceanographic instrument suggest packaging and version control are far more critical than most users or developers in the ocean science community realize. Initial efforts to package feature extraction and skeletonization for a domain scientist lead to the realization that ease of configuration is critical to supporting scientific exploration, experimentation, and illustration. A consideration of the history of marching cubes focuses attention on the gap between the development of methods and the dissemination of fully mature software. These challenges can be framed succinctly as Discovery, Relevance, Adaptability for Ease of Usage, Input/Output Flexibility, Reliability, and Sustainability. The lessons learned here suggest the need for a more sustainable funding model, strong expectations for code dissemination and documentation, attention to the needs of users especially domain scientists, and greater visibility of code development efforts to end users.

DOI: 10.2312/visgap.20201106

1. Motivation

In the last 20 years or so, the visualization research community developed and published significant advances in the ability to visualize, amongst others, three-dimensional, multivariate, and text-based data. Much to the disappointment of the community, the adoption of these visualization methods and tools by their target audiences (domain scientists, operational organizations, and others) has remained slow.

A brief discussion of volume visualization for geoscience illustrates the point. Volume data is data with values at every point in 3D space – or, in practice, data with values at every mesh or grid point within a 3D spatial domain. With advances in modelling and computers, geoscientists studying climate, ocean and earth processes on local, regional and global scales have increasingly turned to three-dimensional modelling to explore complex processes such as ocean circulation, plate tectonics, and weather. Many such processes are fundamentally three-dimensional; that is, the processes behave differently in three-dimensions than in two. Additionally, the collection of oceanographic and geologic data has exploded with the advent of ocean gliders, cabled undersea observatories, and enhanced network capabilities [TWK*19] [KDJ14]. This modelling and data explosion should drive the adoption and exploitation of volume visualization methods. A recent survey of visualization techniques used in ocean sciences shows hundreds of techniques have been developed and applied to ocean data [XLWD19]. However, investigations into the use of visualization tools by oceanographers suggest that expense, difficulties in learning, applicability, and availability have limited adoption of novel visualization techniques [SSVK19] [Edd93].

This paper addresses the challenges of operational and domain usage of visualization methods, especially volume visualization methods in the domain of ocean science. Examples from the author's work in volume visualization of oceanographic data are used to illustrate the issues involved. This concrete context shapes the author's reflections on why so few novel visualization techniques show up in the ocean science community and why so many papers in visualization give little evidence for actual domain science applicability. Although the context is volume visualization, many of the issues are general to all visualization, and even all software development. A framework for discussing the many challenges is presented. Some potential solutions on a community scale are discussed.

2. Case Studies

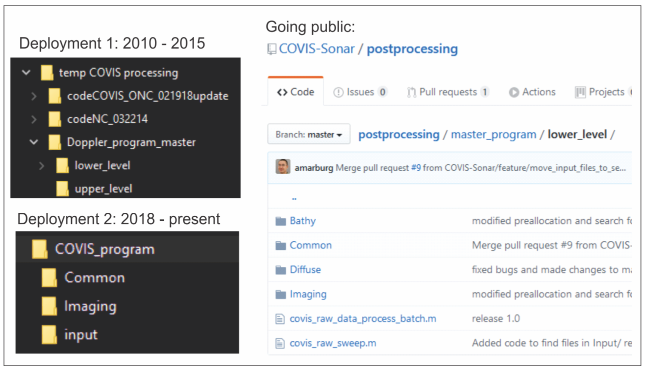
2.1. Feature Tracking – development cycles in an academic lab

In 1996, Silver and Wang [SW96] presented a method to extract features (or objects) from volume data; this method expanded the ideas of Samtaney and others [SSZC94]. Initial applications focused on the merging and splitting of vortices in turbulence simulations [SW97, OM09] and the characterization of bending plumes in underwater acoustic data [RBKHS98]. Later studies have cited or used the algorithms in a wide range of fields from blood flow dynamics [CH02, Jul15] to remote sensing of vegetative variations [Mus12]. Nevertheless, the majority of several hundred citations are from within the visualization community with a minority of citations, including only a few direct uses of the implementations of Silver and Wang, in publications outside the visualization community (that is, in the oceanographic, geoscience or medical literatures). This highlights two gaps between the visualization community and the domain science communities: the gap in adoption of ideas or algorithms and the, even greater, gap in use of implementations.

The Silver and Wang implementation is largely in C++ code, which was initially bound into the Advanced Visual Systems (AVS) environment [AVS] as a user-provided module. Since then, the feature tracking techniques of Silver and Wang [SW97] have undergone many implementations in several different programming environments. The core of feature tracking has continued to be written in C++ while generally leveraging larger visualization environments, such as VisIt [VI2] and Matlab [Mat]. Standalone C++ versions have incorporated varying data input capabilities (rectilinear to unstructured data) and run on various hardware (ordinary desktop to parallel processing). The current version links with various libraries for scientific computing (e.g., VTK [Kit], NetCDF [NC2]) and runs as standalone code. It is independent of any visualization environment (e.g., AVS), but needs additional software (e.g., Paraview [Par] or in-house WebGL viewers) to view the results. The most recent version of the Silver and Wang feature tracking code exists on GitHub [FT1]. These development paths reflect the burden on the lab to support the feature tracking implementation that connects with the burden of adoption placed on the user by requirements and dependencies.

This case study focuses on the burden on the lab, which derives from changes over time and the nature of academic labs. Operating systems have changed in the decades since the feature tracking algorithms were initially developed. Fashions in visualization environments (including third-party libraries and the functionality that they provide) have also changed. Development, adaptation to a specific project, and installation are all eased when using up-to-date dependencies in popular computing environments, but that requires long-term maintenance and development work for an implementation to survive for decades. At the same time, academic labs depend on an ever-changing cohort of graduate students to do the work and this requires ongoing funding to cover stipends and tuition. Furthermore, a graduate student needs to focus on career-building projects (i.e., those leading to publications). Minimal overlap between students and common failures of students to fully document their code, or even leave behind a coherent version of their code, exacerbate the inherent challenges of long-term development and maintenance of software in an academic lab.

For users to adopt the Silver and Wang implementation of feature tracking, that implementation must exist which requires that the lab support it. This example has focused on the burdens placed on an academic lab to develop and support implementations with some mention of the related burdens on the user as software and software environments change over time. The structure of academic labs (turnover of students, need for frequent publications, and dependence on funding) increase the challenges of supporting a particular implementation over time.



amarburg Merge pull request #9 from COVIS-Sonar/feature/move_input_files_to_se...

..	
Bathy	modified preallocation and search f
Common	Merge pull request #9 from COVIS-
Diffuse	fixed bugs and made changes to m
Imaging	modified preallocation and search f
covis_raw_data_process_batch.m	release 1.0
covis_raw_sweep.m	Added code to find files in Input/ re

Figure 2: *Proliferating code versions that all do the same thing, almost but not quite, is a domain scientist's nightmare. A scientist needs to be able to attribute differences in the behavior of the phenomena to actual changes rather than code changes. Incorporating GitHub or other versioning system identifiers into the data or metadata would be one way to improve attribution of changes.*

2.2. COVIS post-processing – an exploration of the challenges of operational software

The Cabled Observatory Vent Imaging Sonar, known colloquially as COVIS, has been deployed on the seafloor under more than 1500 m of water first on the NEPTUNE underwater observatory off British Columbia (operated by Ocean Networks Canada) and more recently at the Ocean Observatories Initiative's Cabled Array off Oregon, USA to monitor the heat output of hot springs rising above underwater volcanoes [XBJ17]. When fully operational, COVIS acquires 8 to 24 data sets per day, each data set forming a 3D volume of backscattering strength (from turbulent fluctuations in temperature in the rising plumes) or a 2D map of thermal variance (from combined diffuse and focused venting). The need to visualize the 3D data has driven a 24-year collaboration between oceanographers and visualization experts.

Visualizing COVIS data requires a specific sequence of processing steps. Datasets retrieved from offshore (via Ethernet and an underwater telecom cable) need to be transformed from raw acoustic data (voltage time series saved as complex waveforms) into beamformed, calibrated, and (usually) gridded data that describes the backscattering strength at points in space. The gridded data can then be directly visualized (isosurfacing and so forth) or further processed to identify features (such as individual plumes) and characterize the static (volumes, centerlines) and dynamic (vertical dilution, temporal evolution) properties.

Algorithmic changes can result from acousticians and geologists discovering better filters, finding errors in calibration, or determining new science goals; they can also result from developments in visualization techniques and implementations. Figure 2 illustrates some of the versioning complexity in software used to process and visualize COVIS data. From a scientific viewpoint, algorithmic changes or other changes in software can create major interpretation issues. Recently, an apparent increase in heat transport turned out to be due to a combination of changes in the data processing and sonar settings. Transitioning code tracking to software versioning packages helps (the COVIS team has begun a GitHub repository [COV]). Incorporating the GitHub identifier within the metadata attached to the data will be key increasing interpretability of changes in phenomena.

Operationally, the COVIS data processing pipeline happens 365 days a year, 8 to 24 times a day (when the file arrives on shore) as well as on demand (when a user asks for all the data between Mar 2012 and Jan 2014 or algorithmic changes drive reprocessing all existing data). Observatory data portals serve raw data, gridded data and visualization products to the ocean science community. Operational use demands robust, scriptable software that can handle data glitches while running autonomously, but does not need a GUI or complex user interface.

The acoustic instrument and technique development and the oceanographic interpretation and experimentation impose constraints on the visualization developments. MATLAB is the software environment of choice for acoustics, because of its strong signal processing capabilities. The COVIS team largely uses a mixture of Microsoft Windows and Mac OS based computers. So, despite the long-term collaboration between the COVIS team and the Silver lab, routine visualization of COVIS data is still primarily done in MATLAB. This suggests that for the COVIS team to adopt feature tracking, or any other visualization software, the software needs to be integrated with MATLAB or be able to run independently on a variety of platforms.

For the visualization community, these are the key lessons in this case study: (1) Choices about platforms, languages, and ease of use will impact how likely a domain scientist or science community is to adopt a software implementation at least as much as the effectiveness of the visualization. (2) Scientists, on any scale, do not make clear common choices in platforms etc. indicating that interoperability should be a major consideration. (3) The adoption of visualization techniques requires that someone invest the time to implement them in readily usable form. Finally, this scenario also emphasizes that tracking software versions is actually important to domain science (even if scientists are bad at it) and efforts to imprint such metadata into figures directly should be applauded.

2.3. VSK – issues with exploring plume bending

A key aspect of understanding the dynamics of plume-ocean interactions is finding the centerline of plume rise. Post-processing of the COVIS data in the previous section includes several slice-by-slice methods of centerline extraction that rely on a pre-specified region to identify the plume of interest, especially when multiple plumes are present in the data. In contrast, feature tracking's extraction capabilities could characterize multiple plumes in a single pass while more robustly identifying the plumes present. In an effort to apply feature extraction and skeletonization techniques to the COVIS data, the feature extraction capability of feature tracking was combined with an implementation of skeletonization and compiled for use within MATLAB, where it is known as VSK [VSK]. The advantage of MATLAB is that input data remains in its native format and the feature tracking function can be included seamlessly in the COVIS processing sequence or run on demand.

At least, that was the theory. In practice, the VSK code has not

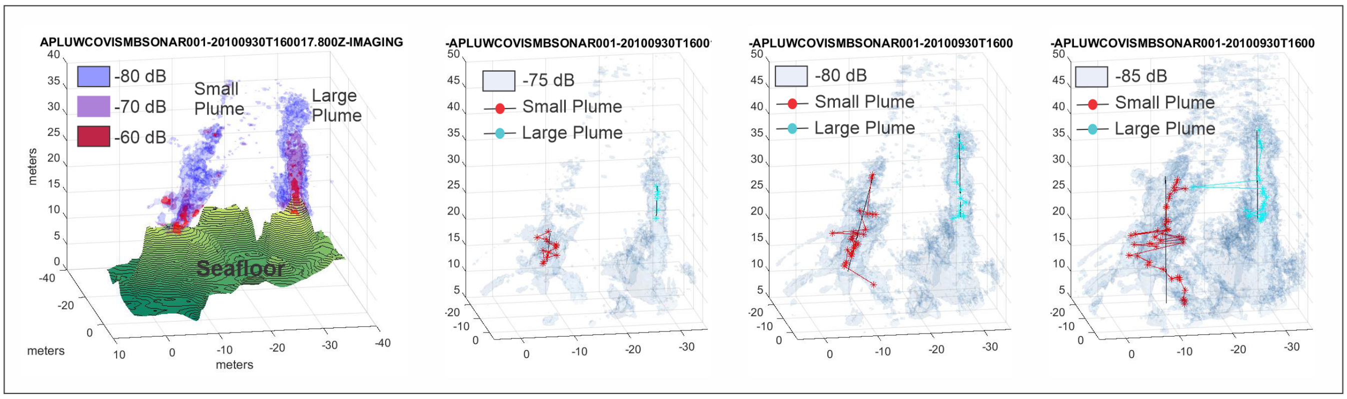

APLUWCOVISMBSONAR001-20100930T160017.800Z-IMAGING
-80 dB
-70 dB
-60 dB
Small Plume
Large Plume
Seafloor
meters
meters
meters
APLUWCOVISMBSONAR001-20100930T1600
-75 dB
Small Plume
Large Plume
APLUWCOVISMBSONAR001-20100930T1600
-80 dB
Small Plume
Large Plume
APLUWCOVISMBSONAR001-20100930T1600
-85 dB
Small Plume
Large Plume

Figure 3: *Feature extraction at multiple thresholds applied to COVIS data [COV20]. While the orientation of the larger rising plume is revealed at all three thresholds, the smaller plume and the plume-rock base connections only become apparent at the lower threshold values. Backscattering strength is given in decibels (dB) relative to the output signal. Seafloor is an independent data set [CCT*08].*

proved useful. First, the threshold used to define the features is buried in the middle of the code. Second, the user interface focuses on a choice of multiple techniques and requests the user to find and load data via browsing. Third, the output focuses on producing only the orientation of the centerline (azimuth and inclination or visually a line) rather than the full structure of the plume (the actual skeleton itself).

Had the development of VSK started with a design study, at least two distinct tasks or use cases could have been discovered: (1) an early experimental phase wherein the scientist explores the parameter space and (2) a routine analysis phase where long time series of data would be run autonomously with a standard set of parameters. Each phase places different design requirements on potential feature extraction, tracking, and skeletonization software.

During the experimental phase, the objective is to explore the useful feature-defining threshold space as well as other parameters (thinning and connectivity for skeletons). The backscatter range of COVIS data can span about five orders of magnitude as the temperature fluctuations in plumes decrease exponentially with dilution (see Figure 3). Generally, low threshold values represent ambient ocean, middle values the plume core, and higher values rock, but the actual values can vary over time and space due to noise, calibration shifts, and actual changes in phenomena. Experimentation with differing threshold values (Figure 3) is key to establishing an accurate understanding of the true transfer function. The process of experimentation involves working with the same small number of files to establish a best-fit threshold and perhaps adjust the data preprocessing steps. During the experimental stage, a well-designed software interface would enable the constant setting of one or a small number of files but enable rapid and frequent changes in parameters.

In contrast, the routine analysis phase involves the processing of a long time series of data files (>5000 files in the COVIS case) either all at once or as new data is acquired. In this case, the parameters will be the same for every file. Nevertheless, a well-designed software interface should still separate parameters from the code and support a simple method of feeding in files.

This scenario highlights the importance of design studies to identify the scientific tasks when software is developed with a particular collaboration or for a particular domain; the intended tasks will set the requirements for flexibility and convenience in input specifications. The exploration of data (where a small number of files is viewed) will drive distinct requirements to the exploitation of data (where many files need to be processed). Software developed on a more generic basis, where specific use cases may be hard to determine, should anticipate that libraries of functions are more readily incorporated into workflows than full-scale systems.

2.4. Considering Marching Cubes

Since the marching cubes algorithm for extracting an isosurface from three-dimensional volume data was published in 1987 [LC87], a wide variety of implementations and improvements have been published (see Newman and Yi [NY06] and Custodia and others [CPS19] for recent reviews). However, commonly, these advances exist as one-off implementations for the publication or incomplete prototypes. The use of these or similar isosurfacing techniques have made it into some widespread software environments, but some of the most accessible and popular software environments still lack easily-incorporated isosurfacing capabilities.

Experts in visualization or programming can certainly find source code or packages in whatever system they prefer and have the skills to create their own implementations. However, domain specialists are often novices when it comes to visualization and maybe even software in general. Ocean operations specialists (e.g., those running an oceanographic observatory) may be more interested in libraries of functions that integrate with existing tools or cleanly packaged executables. More importantly, such end users do not generally connect what they are using in commercial visualization environments with the technique development efforts made in the visualization community (especially as commercial software rarely reveals what, if any, connection exists).

Two concerns should inform discussion in the visualization community. One is how do end users find software. Relying on word-of-mouth, random google searches, and visualization conference attendance will not garner many users. The other is what credit should apply to academic development of techniques (largely existing in and on publications and citations) as opposed to the proprietary development of commercial tools.

3. Building a framework

To build a framework to discuss the challenges in broad adoption of new visualization techniques, this paper started with a series of specific issues that come from both conversations with domain users and software developers and from the output of a series of workshop on the gaps between development and adoptions in visualization [RCM*20]. The first two columns of Table 1 summarize the issues in the case studies reviewed above.

From the lessons highlighted in these case studies, we can form a framework for discussion highlighting the broad areas of Discovery, Relevance, Adaptability of Code for Usage, Input/Output Flexibility, Reliability, and Sustainability. The discussion in the next section covers why each of these areas is important and uses the case studies above to illustrate key ideas.

4. Reflecting on Challenges to Adoption

4.1. Discovery

Section 2.4 (Marching Cubes) highlighted issues of the visibility of visualization tools and techniques, especially outside the visualization community. **Discovery** relates to how users find tools and techniques. For publications, there are search tools like Google Scholar and Web of Science. But most often users learn of software and visualization techniques by word-of-mouth or happenstance. Either they saw it in a talk or a paper or it came up in a conversation or a professor used it in a course. The EarthCube initiative has started a list of visualization (and other) tools useful to geoscientists but at present does not provide a comprehensive survey [Ear].

Another aspect of discovery happens when the user finds too many tools, especially if each requires some level of learning or workflow modification. The user needs to weigh the virtues of a new visualization technique against the effort to install, learn, and use. Scholarly surveys often tout the virtues of techniques but give little information on how to find implementations or even if they exist. In contrast, web-based searches discover more irrelevant or inapplicable tools than useful ones, especially if the user doesn't know that to get fully 3D visualization tools the search term needs to be "volume visualization" rather than "3D visualization".

4.2. Relevance

Sections 2.3 (VSK) and 2.4 (Marching Cubes) highlighted the importance of design studies and making explicit design choices based on the anticipated end users and use cases. **Relevance** summarizes this connection between user tasks and software specifications, incorporating an inherent consideration of the intended users' needs. Attendance by the author at visualization conferences led to the casual observation that few scientific visualization presentations spend significant time assessing the scientific value (for insight or illustration) of their visualizations.

Two specific aspects can illustrate why failing to align software design with user tasks and needs can create real issues for domain scientists. First, science is fundamentally an experiment-driven field. One example is never enough: a process needs to be repeated with slight changes in input many times to describe the full breadth of a phenomenon (Figure 3). Second, creating informative illustrations involves more than aesthetic considerations. For a domain scientist to want to expend the time and effort to adopt visualization methods into their workflow, the visualizations need to illustrate a key phenomenon using their data, provide a (visual) metric for the phenomenon, or increase scientific understanding of the phenomenon (Figure 4).

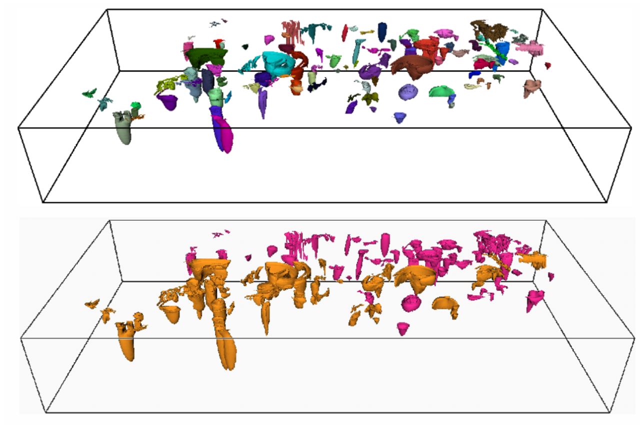


Figure 4: *Randomly colored eddies (top) convey only their basic spatial distribution unlike eddies colored by spin direction (bottom), which give a sense of the distribution of energy and transport as well. Images based on Liu and others [LSB19].*

4.3. Adaptability of Code for Usage

Sections 2.1 (Feature Tracking) and 2.2 (COVIS operations) highlight the efforts involved in integrating visualization techniques, libraries, and tools into existing workflows and the documentation needed to make such integration possible. **Adaptability of Code for Usage** encompasses ease of use and installation issues from dependencies and packaging to coding environments and the need for adequate documentation. For domain scientists, wishing to visualize their data, the time investment in finding a suitable system, upgrading it to have the right packages, and then compiling the visualization tool can exceed the utility of the visualization results. Good documentation, specifying dependencies and clarifying use can help ease the burden. Direct collaboration or libraries that can be integrated readily into existing workflows are alternatives. In the end, either the visualization or the domain scientist will need to invest the time and effort.

Case Study	Specific Issue	Challenge Area
Feature tracking (Sec. 2.1)	Long-term support of implementations in the face of changing software environments	Sustainability
	Supporting user expectations for ease of installation and use; Documenting installation procedures, use cases, and changes	Adaptability for Ease of Use
	Difficulties discovering or choosing software	Discovery
COVIS operations (Sec. 2.2)	Integrating visualization software into an existing workflow; Developing suitable software requirements and dependences for user groups with diverse computing environments	Adaptability of Code for Usage
	Validating that apparent visual changes are real rather than software changes, especially from data output after the fact; Coping with non-standard and potentially inconsistent input data	Reliability
VSK (Sec. 2.3)	Utilizing effective design studies to determine anticipated user tasks for data exploration, data exploitation or generic use	Relevance
	Designing software to task requirements, such as including appropriate configuration structure to support experimentation	Input/Output Flexibility
Marching Cubes (Sec. 2.4)	Choosing appropriate implementations for desired end users	Relevance
	Creating visibility for techniques and implementations beyond the visualization community; Connection between commercial packages and formal visualization techniques obscure or non-existent	Discovery

Table 1: *Organization of issues arising in the case studies*

4.4. Input/Output Flexibility

Section 2.3 highlights design considerations such as what configuration structure will best support the users' tasks from exploration to experimentation to exploitation. **Input/Output Flexibility** deals with the flexibility to work with multiple inputs and outputs, the importance of recording metadata and algorithmic assumptions, and (again) the need for complete documentation. In moving from prototypes designed and used by students in computer science/engineering to operational or routine tools used by geologists or oceanographers, there is a need to adjust how the human interacts with the software.

End users should not need to recompile source code to change a threshold value or the location of the input data files. Scriptability is a term that summarizes how well all the I/O (Input and Output) of a particular software can be described external to the program. Documentation is an important aspect of flexibility. It tells the user how to change the input information, where the output information goes, and information needed to interpret the output.

4.5. Reliability

Section 2.2 (COVIS operations) highlights the importance of validating scientific information and touches on operational issues of consistency and procedural testing. **Reliability** is about having confidence: confidence that the code will run (especially important in operational settings) and confidence that the output is correct and can be interpreted. Addressing reliability means discussing test procedures, assumptions about data, consistency of output, and validation of results.

Scientists think about validation of results in terms of assessing the correctness of the output and asserting the relevance of the test data to real world data. Real world data is noisy and messy, can vary in format and sampling details, and include unexpected situations. If the testing doesn't include complicated enough data, data with enough noise, or appropriate edge cases, the actual output with real world data may be unpredictable or inappropriate.

4.6. Sustainability

Section 2.1 highlights the burdens long-term development and support of implementations place on academic labs. **Sustainability** speaks to the domain scientists' (and other end-users') need for visualization software to keep working over decades, decades in which software environments, operating systems, and computer hardware keep changing. Individual labs or researchers struggle to find funding and time to keep making the (small) changes in implementations needed to keep code viable within an academic funding model driven by continued publication of novel techniques and applications. Even long-term serving of a stable implementation can be a challenge, although GitHub [Git] has certainly helped here.

Another challenge of sustainability in academic labs is the irregular turnover of graduate students, resulting in gaps and failures to transfer knowledge. The result for the Silver and Wang feature tracking is powerful code for tracking features in volume data that is underutilized due to issues in documentation, packaging, and installation. This study actually took a first-time look at citations to see how many users there are and who they are; the results suggest that most users are within the visualization community or that domain scientists do not consistently or appropriately cite implementations.

5. Concluding thoughts on solutions

Using four case studies, this paper considers why the adoption of volume visualization software by domain scientists (especially in ocean science) has lagged behind the development of visualization techniques. While many of the issues discovered are well-known issues for any software development project, they still form barriers to adoption of visualization techniques and tools. Other issues are specific to the needs and capabilities of the ocean science community. In the end, a visualization developer will weigh the costs of developing for the ocean science community (greater time invested in development) against the benefits (collaborations, novel applications, visibility outside of the visualization community).

Reflections on the commonality between case studies suggests that these challenges can be framed as challenges of Discovery, Relevance, Adaptability of Code for Usage, Input/Output Flexibility, Reliability, and Sustainability. This framework provides some structure for exploring potential solutions.

Responding to challenge of Discovery requires reaching out beyond the visualization community. Pre-print servers and code repositories increase access to algorithms, code and implementations, but do not necessarily make them easier to find, especially if the potential user has limited knowledge of the appropriate keywords. One useful direction is the recent efforts of EarthCube in the geoscience community to combine inventories of tools (and data) with the infrastructure needed to use them [Ear]. Visualization developers interested in ocean or earth science users may wish to become involved with that effort. The visualization community could consider an inventory effort of a broader nature. Greater visibility to users outside the visualization community might also motivate greater investment into usability.

By and large, the visualization community knows how to respond to the challenge of Relevance: there is extensive literature on how to conduct a design study already (e.g., [Mun14]). Even student projects should incorporate a discussion early on of the intended end users' tasks and eventual scope: whether the main focus is technique creation and whether an implementation will be used only in a visualization research lab, is intended as a generic tool, or is part of a collaboration with domain scientists.

Responding to the challenges of Adaptability of Code for Usage and Input/Output Flexibility also hinges recognizing the intended end-users. Development of visualization techniques and tools for domain science needs to recognize the capabilities of the domain scientist: Ocean and geo scientists are generally better at using software than developing software, so efficiency suggests the investment into usability will fall on the development or visualization side. Communicating what is a key input or key knowledge for interpretation is a critical part of the conversation between scientist and developer. However, as the case studies above show, while long-term collaborations build understanding of what can be visualized for the domain scientist and what should be measured for the computer scientist, they do not inevitably lead to convenient configuration or greater usability. Community or journal expectations of code and documentation in repositories might encouraging better attention to documentation.

Responding to the challenge of Reliability requires considering the specific nature of real data (messy and full of gaps) as well as following thorough with testing, documenting, and validating results. Teaching software engineering early in the training of graduate students pursing visualization could be a successful community response to the interrelated challenges of Relevance, Adaptability of Code for Usage, Input/Output Flexibility and Reliability.

The challenges of decadal scale development and support weave their way through much of this narrative. In the case study on feature tracking, a more sustainable funding model would potentially increase the overlap between students leading to less wasted development efforts. Within academia, turnover of developers, the need for novelty in order to publish, and the lack of academic recognition for the work of packaging and supporting software underlie much of this conversation.

Responding to the challenge of Sustainability will take change and effort from both the visualization and domain science communities. On the one side, the visualization communities could promote professional standards for packaging, documenting, and testing software (including buy-in by the academic community to teach such standards) and change expectations for best practices. On the other side, the domain science communities could better document their usage of both software packages and novel visualization techniques providing the tracking needed to establish which lines of visualization research were valuable to the science community.

Making technique and software usage more visible to both the domain science and visualization communities may be a critical step in building funding agency support for the development and maintenance of widely used tools. Routine citations of software used in domain scientists' publications would help establish the scale of the user community. Perhaps software needs a DOI-like identifier, something that is being adopted for physical rock samples and various databases [DKT*17] [LKA*11]. A clearer picture of usage might also help rebalance the visualization community's emphasis on publishing novelty for novelty's sake with a strong sense that novelty is of most value when an end-user community finds the novelty of value in their own applications.

References

[AVS] Advanced visual systems. `http://www.avs.com`. Last accessed April 21, 2020. 2

[CCT*08] Clague D. A., Caress D. W., Thomas H., Thompson D., Calarco M., Holden J., Butterfield D.: Abundance and distribution of hydrothermal chimneys and mounds on the endeavour ridge determined by 1-m resolution auv multibeam mapping surveys. In *AGU Fall Meeting Abstracts* (December 2008). 4

[CH02] Clayton R. H., Holden A. V.: A method to quantify the dynamics and complexity of re-entry in computational models of ventricular fibrillation. *Physics in Medicine & Biology 47*, 2 (2002), 225. 2

[COV] Covis data processing software. https://github.com/COVIS-Sonar. Last updated April 22, 2020. 3

[CPS19] Custodio L., Pesco S., Silva C.: An extended triangulation to the marching cubes 33 algorithm. *Journal of the Brazilian Computer Society 25*, 1 (2019), 1–18. 4

[DKT*17] Devaraju A., Klump J., Tey V., Fraser R., Cox S., Wyborn L.: A digital repository for physical samples: concepts, solutions and management. In *International Conference on Theory and Practice of Digital Libraries* (September 2017), Springer, pp. 74–85. 7

[Ear] Earthcube tools inventory. https://www.earthcube.org/tools-inventory. Last accessed April 24, 2020. 5, 7

[Edd93] Eddy W. F.:. In *Statistics and Physical Oceanography*. National Academies, 1993, ch. 6. URL: https://www.nap.edu/read/9028/chapter/6. 2

[FT1] Feature tracking implementation aka silver and wang. https://github.com/liuli4016/feature_tracking. Uploaded Nov. 1, 2018. 2

[Git] Github. https://github.com/. Last accessed April 24, 2020. 6

[Jul15] Julian M.: *Illustration-inspired visualization of blood flow dynamics*. MAS thesis in mechanical and industrial engineering, University of Toronto, 2015. 2

[KDJ14] Kelley D. S., Delaney J. R., Juniper S. K.: Establishing a new era of submarine volcanic observatories: Cabling axial seamount and the endeavour segment of the juan de fuca ridge. *Marine Geology 352* (2014), 426–450. 2

[Kit] Vtk – the visualization toolkit, kitware. https://vtk.org/. Last accessed April 24, 2020. 2

[LC87] Lorensen W. E., Cline H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics 21*, 4 (1987), 163–169. 4

[LKA*11] Lehnert K., Klump J., Arko R. A., Bristol S., Buczkowski B., Chan C., Chan S., Conze R., Cox S.: Igsn e.v.: Registration and identification services for physical samples in the digital universe. In *American Geophysical Union, Fall Meeting; 2011-12-5/9; San Francisco* (2011), AGU. doi://hdl.handle.net/102.100.100/102094?index=1. 7

[LSB19] Liu L., Silver D., Bemis K.: Visualizing three-dimensional ocean eddies in web browsers. *IEEE Access 7* (2019), 44734 – 44747. 5

[Mat] Matlab, desktop environment for analysis. mathworks. https://www.mathworks.com/products/matlab.html. Last accessed April 24, 2020. 2

[Mun14] Munzner T.: *Visualization analysis and design.* CRC Press, 2014. 7

[Mus12] Musaningabe C. G.: *Assessing vegetative drought from multi-temporal NDVI images*. Phd, University of Twente, 2012. 2

[NC2] Netcdf, software libraries for array-oriented scientific data. unidata program center. https://www.unidata.ucar.edu/software/netcdf/. Last accessed April 24, 2020. 2

[NY06] Newman T. S., Yi H.: A survey of the marching cubes algorithm. *Computers & Graphics 30*, 5 (2006), 854–879. 4

[OM09] O'Farrell C., Martin M. P.: Chasing eddies and their wall signature in dns data of turbulent boundary layers. *Turbulence* (2009). 2

[Par] Paraview, open source analysis and visualization software, kitware. https://www.paraview.org/. Last accessed April 24, 2020. 2

[RBKHS98] Rona P., Bemis K., Kenchammana-Hosekote D., Silver. D.: Acoustic imaging and visualization of plumes discharging from black smoker vents on the deep seafloor. In *Visualization'98. Proceedings* (1998), IEEE, pp. 475–478. 2

[RCM*20] Reina G., Childs H., Matkovic K., Bühler K., Waldner M., Pugmire D., Kozlíková B., Ropinski T., Ljung P., Itoh T., Gröller E., Krone M.: The moving target of visualization software for an increasingly complex world. *Computers & Graphics* (January 2020). 5

[SSVK19] Stocks K. I., Schramski S., Virapongse A., Kempler L.: Geoscientists' perspectives on cyberinfrastructure needs: A collection of user scenarios. *Data Science Journal 18*, 1 (2019), 1–15. doi://doi.org/10.5334/dsj-2019-021. 2

[SSZC94] Samtaney R., Silver D., Zabusky N., Cao J.: Visualizing features and tracking their evolution. *Computer 27*, 7 (1994), 20–27. 2

[SW96] Silver D., Wang X.: Volume tracking. In *Proceedings of Seventh Annual IEEE Visualization'96* (October 1996), IEEE, pp. 157–164. 2

[SW97] Silver D., Wang X.: Tracking and visualizing turbulent 3d features. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (1997), 129–141. 2

[TWK*19] Trowbridge J., Weller R., Kelley D., Dever E., Plueddemann A., Barth J. A., Kawka O.: The ocean observatories initiative. *Frontiers in Marine Science* (2019). 2

[VI2] Visit, open source visualization tool. https://wci.llnl.gov/simulation/computer-codes/visit/. Last accessed April 24, 2020. 2

[VSK] Vsk, a skeletonization implementation by li liu. https://github.com/liuli4016/VSK. 3

[XBJ17] Xu G., Bemis K., Jackson D.: Sounding the black smoker plumes. *EOS* (November 2017). doi://doi.org/10.1029/2017EO086289. 3

[XLWD19] Xie C., Li M., Wang H., Dong J.: A survey on visual analysis of ocean data. *Visual Informatics 3*, 3 (2019), 113–128. 2