Dynamic Collaborative Visualization Ecosystem (DynaCoVE)

Christopher Koehler, Andrew Berger, Raksha Rajeshekar, Thomas Wischgoll, *Member, IEEE*, and Simon Su

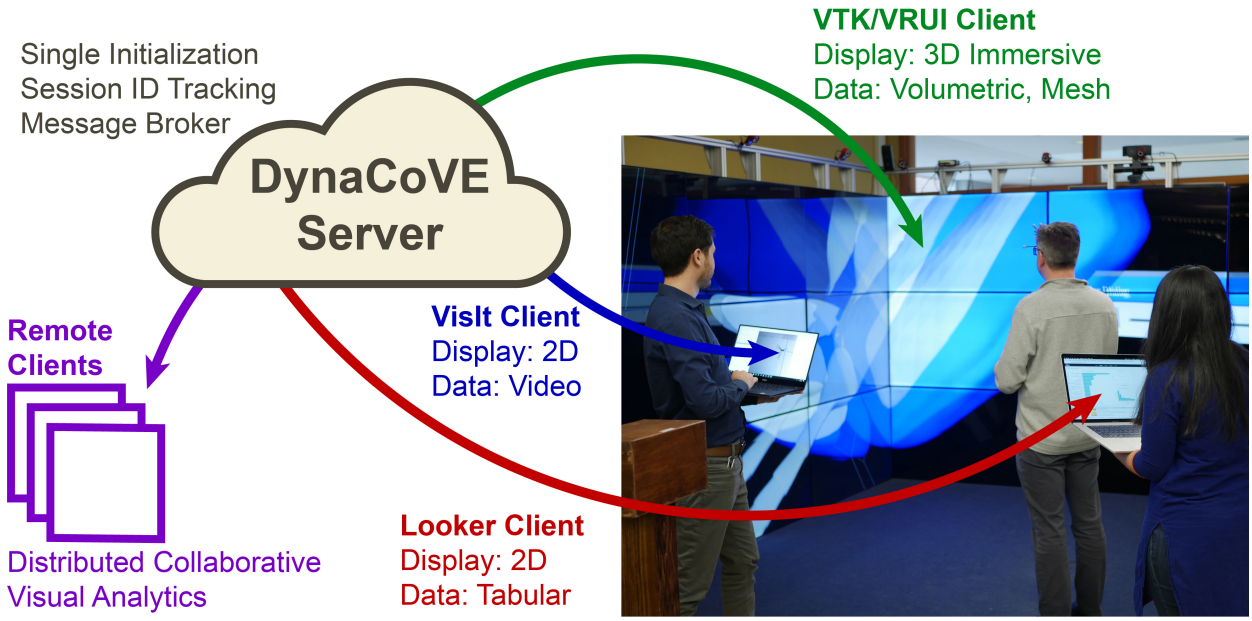


Fig. 1. Collaborative visual analytics across display technologies with DynaCoVE. Client wrappers encapsulate standard visualization tools that were not originally capable of interacting together. Supplemental illustration denotes components at play in an actual session.

Abstract— There is no one display device or software package that is ideally suited for interactively visualizing related non-spatial, 1D, 2D, 3D, and 4D datasets. This is a major drawback, as the benefits of interactive visualization and advanced display technologies cannot be brought to bear. The Dynamic Collaborative Visualization Ecosystem (DynaCoVE) framework addresses this limitation by unifying SciVis, InfoVis, and display technology tools. Pre-existing packages are wrapped as DynaCoVE clients. This entails tracking user interactions with their GUI's and also automatically updating them in response to cues from the DynaCoVE server. The DynaCoVE server then acts as a message broker between many collaborating clients. A messaging protocol was defined based on ZeroMQ's majordomo protocol to enable efficient communication between clients and the server. Thus, visualization software packages that were never intended to be used together can be linked to perform cross display visual analytics. The system is intended to be data and user centric while remaining software, algorithm and display agnostic. This is accomplished in part by providing a common meta-visualization graph interface to setup cross-display visualizations. To date DynaCoVE clients for Looker, VisIt, and VTK have been tested together with traditional monitors, tiled displays, and an immersive CAVE-type system.

Index Terms—Data fusion and integration, large and high-res displays, collaborative and distributed visualization, integrating spatial and non-spatial data visualization, coordinated and multiple views.

1 Introduction

Data is rapidly getting larger and more diverse. Interactive, human-in-the-loop visualizations linking all data sources pertaining to the most challenging problems is more advantageous than static visualizations of individual pieces of data in isolation [1]. However, the software needed to interactively bring multiple state-of-the-art visualization tools and display hardware technologies to bear together is sorely lacking.

For example, if a user is interacting with 4D simulation data in a Cave environment and they find something interesting they may wish to interactively bring up the contextual information surrounding that instant on an adjoining wall display for a collaborator to visualize using an appropriate tool. Similarly, interacting with the contextual data should be able to drive the 3D volumetric display. This should not be complicated! However, it is potentially not even possible without employing a slew of one-off tools due to a lack of software support.

The Dynamic Collaborative Visualization Ecosystem (DynaCoVE) framework was created to address these limitations. It provides an efficient framework to build interactive visual analytics tools from a suite of pre-existing components. Thus, the aim of DynaCoVE is not to replace the current state-of-the-art visualization tools but rather to enhance them with increased potential for collaborative interaction and

- *Christopher Koehler is with Universal Technology Corporation. E-mail: ckoehler@utcdayton.com.*
- *Andrew Berger is with Universal Technology Corporation. E-mail: aberger@utcdayton.com.*
- *Raksha Rajeshekar is with Wright State University. E-mail: rajashekar.3@wright.edu.*
- *Thomas Wischgoll is with Wright State University. E-mail: thomas.wischgoll@wright.edu.*
- *Simon Su is with the US Army Research Laboratory. E-mail: simon.m.su.civ@mail.mil.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

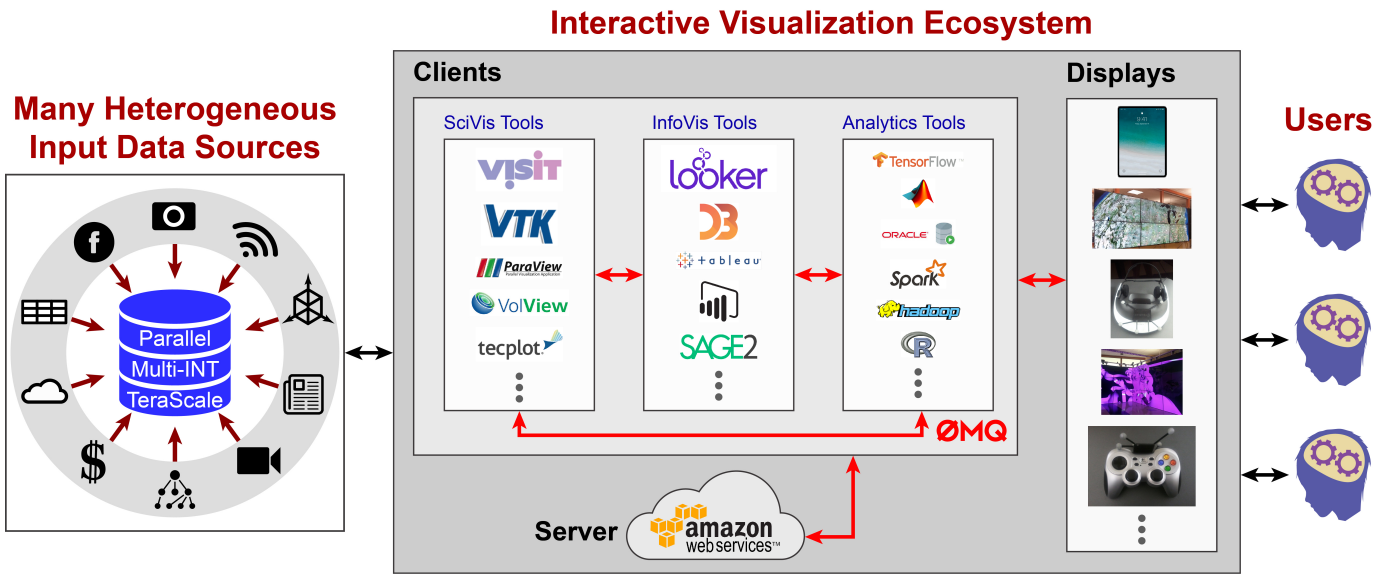


Fig. 2. IIllustration of the visualization ecosystem concept and cross section of the many components that are envisioned for eventual inclusion. DynaCoVE currently has Looker, VisIt, and VTK/VRUI clients. More are in the process of being added.

greatly increased display realestate. An example of a visual analytics solution created with DynaCoVE is shown in Fig. 1.

2 Related Work

The current work is related to several branches of previous research including interactive information and scientific visualization, frameworks for collaborative visual analytics, and methods of enabling interaction between monolithic multidisciplinary analysis tools that were not originally intended to be used together. An overview of this previous work is presented now.

Visual Analytics, the science of human reasoning supported by interactive visual interfaces, is a broad field [2, 3]. It encompasses elements of visualization, data mining, statistics, big data, and human-computer interaction. Speaking purely on the visualization size, interactive visual analytics is most pervasive in information visualization (InfoVis) type tools and methods that are primarily appropriate for 2D displays. This can be seen in current commercial tools, such as Looker [4], Tableau [5], and Power BI [6], which provide a suite of basic InfoVis constructs. Such tools provide simple, intuitive interaction paradigms such as linked views, selection, and brushing that are easy to use and understand. From an interaction standpoint, DynaCoVE is initially targetting similar capabilities, while providing interaction across different packages and display environments.

User interaction has also been demonstrated for isolating important regions in scientific visualization. For example, Doleisch developed SimVis, which uses traditionally infovis techniques for interactive scientific visualization [7]. Shi et al. extended this approach by incorporating pathline attributes into a similar system [8]. Muigg et al. used interactive linked views with three kinds of focus [9]. Such methods leave finding important regions to users by providing tools to help them isolate their domain of interest. Koehler et al. used interactively controlled adjoint enhancement to adjust emphasis of physical regions based on their importance relative to a specific quantity of interest [10].

More recent efforts have addressed some aspects of a visualization ecology [11] such as collaborative visualization, mixed spatial and nonspatial visualization, and hybrid display visualization systems. For example Su et al. created a hybrid scientific and non-scientific visualization system [12]. Marrinan et al. developed SAGE2 [13] to facilitate collaboration on large shared displays. Kobayashi et al. created ParaSAGE [14], an extension of ParaView to SAGE2. There has also been some work on a collaborative high resolution data visualization framework [15] on top of the ParaViewWeb framework [16]. Despite these advances, to the best of our knowledge no current tool takes interactive visualization to the level of cross datatype, cross visualization software package, and cross display hardware interactivity that is targetted by DynaCoVE.

Implementing a visualization ecosystem also shares common challenges with other efforts in terms of linking software packages and translating from standard configuration formats to drive a multitude of tools. One such effort is the Computational Aircraft Prototype Synthesis (CAPS) program [17]. Here the goal is to perform multi-disciplinary simulations from a common geometry. This requires initializing many analysis tools from a universal configuration format. DynaCoVE's interface must do something similar in terms of initializing many visualization tools from a common interface. On the communication side the Hermes framework for cross-language remote procedure was developed to simiplify interacting between analysis components in a multidisciplinary simulation running on separate machines [18].

3 Background

Other disciplines including data science and machine learning are recently discovering that a multiple intelligence (multi-INT) data fusion approach is capable of achieving results beyond traditional 'stoved-piped' processing of individual data sources [19–21]. Interactive visualization stands to benefit from a more unified approach as well.

However, interactive visualization of many related but differently typed datasets across different 2D, 2.5D, and 3D displays is currently a very cumbersome process. This is partially due to the fact that no single software package or display/input device is ideal for the many types of non-spatial, 1D, 2D, 3D, and 4D data available, and partially due to a lack of suitable software support. Interactive visualization across the traditional visualization disciplines (InfoVis and scientific visualization) is particularly lacking.

The solution is not to create a monolithic piece of software that will 'do everything'. Instead, the solution is a visualization ecosystem [11], which modularizes and unifies existing capabilities from the fields of visual analytics, InfoVis, data analytics, big data, scientific visualization and display hardware. Existing state-of-the-art tools are modularized and unified in a composable, scalable, data and user centric interface. Visual analytics solutions can then be created on the fly from combinations of the tools that have been wrapped into the ecosystem.

The Dynamic Collaborative Visualization Ecosystem (DynaCoVE) is a fully functioning visualization ecosystem. It has been demonstrated with multiple state-of-the-art InfoVis and scientific visualization tools that were not originally designed to interact together, including VisIt [22], Looker [4], and VTK/VRUI [23, 24]. Further, DynaCoVE's

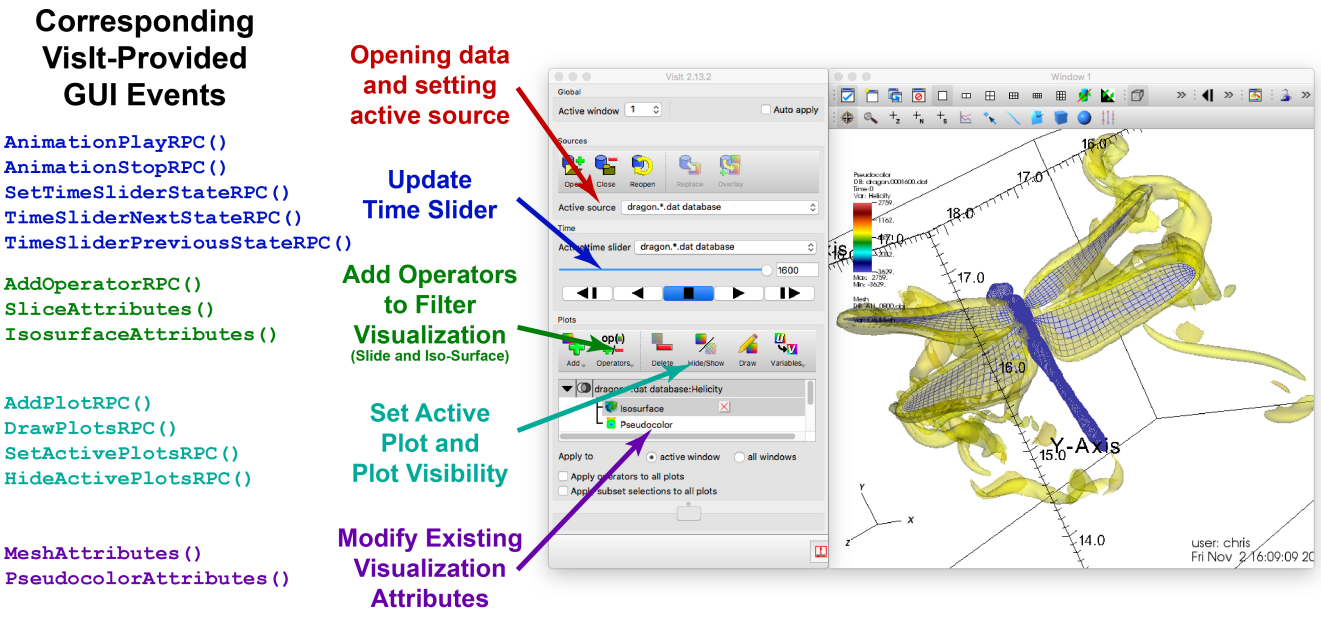


Fig. 3. Illustration of VisIt Components that can drive and receive interactions from other DynaCoVE clients

architecture was designed to be rapidly extended beyond the initial toolset by providing standard client stub interfaces in multiple languages to wrap additional tools. Thus, the system remains agnostic to the individual visualization tools, algorithms and display technologies. The concept is illustrated in Fig. 2.

4 Method

To create interactive visualizations between pre-existing visualization packages they are first encapsulated as DynaCoVE clients. A DynaCoVE server then acts as a message broker between clients. There is currently a local version of the server as well as a public Amazon Web Services (AWS) version. The ZeroMQ distributed communication library handles message passing between the clients and the server. This section presents more detail on each of these components.

4.1 DynaCoVE Clients

A visualization ecosystem requires that existing tools be encapsulated in full or in part to operate as clients in a larger interactive system. These tools may be large and monolithic in nature, however there is a simple test to determine how well they can function in a visualization ecosystem. In order to have interactions flowing in and out of any given client, a package must meet the following two criteria for a fully functional inclusion in DynaCoVE.

1. One can catch and expose a desired subset of user interactions with its visual interface to be sent to the DynaCoVE server to drive updates to visualizations in other displays.
2. Its visual interface can be updated through an API in response to instructions incoming from the DynaCoVE server due to to users interacting with other clients.

At this time, DynaCoVE has working clients for VisIt, VTK/VRUI, and Looker.

4.1.1 Visit Client

VisIt is a monolithic, distributed, parallel, scientific visualization tool. It has the ability to utilize parallel capabilities of modern graphics hardware and HPC resources, and it can visualize data without needing to move it around. A DynaCoVE client has been created for visit.

There are quite a few events that the VisIt GUI can generate in response to a user's interactions with it, as can be seen in the VisIt Python Manual [25]. Using custom callback functions registered for certain relevant GUI events, the DynaCoVE VisIt client is able to expose the necessary interaction history to meet the first criterion. The DynaCoVE client currently has callbacks to capture VisIt GUI events from the following UI components, their corresponding VisIt GUI events are shown in Fig. 3.

- Time Slider
- Data Plots
- Data Filters
- Visualization Attributes

With regard to the second criterion to be a DynaCoVE client, VisIt's Python command line interface (CLI) was implemented such that existing Python scripts can be augmented with code to control essentially any aspect of VisIt that its GUI can control. Thus, VisIt can be made to respond to user interactions with other tools. The portions of VisIt's GUI that can be driven from external packages or used to drive external packages is shown in Fig. 3.

4.1.2 VTK/VRUI Client

A VTK/VRUI [23, 24] DynaCoVE client was also created. This client is more low level in nature than the VisIt client, however it offers stronger scalability to different display and input device types. It builds on the VRUI environment to support different display systems ranging from conventional desktop environments to full-scale CAVE-type display systems without requiring any additional work. It also supports different types of head-mounted displays that are fully tracked for more sophisticated input paradigms, as well as touch-based input devices. The VTK/VRUI client current capabilities include support for unsteady 3D volumetric, vector field and mesh data. It can receive instructions from the server to change data sets, change time steps, and change iso-values. Given the low level nature of this client, it was straightforward to build the necessary interface event logging straight into the C++ source code.

4.1.3 Looker Client

The DynaCoVE Looker client relies primarily on Looker's Data Actions to drive communication with the server. Data Actions can be defined on any Looker data model to allow users to perform tasks in other tools from within Looker. This functionality was a major factor in Looker's early inclusion in DynaCoVE. When a user initiates a Data Action

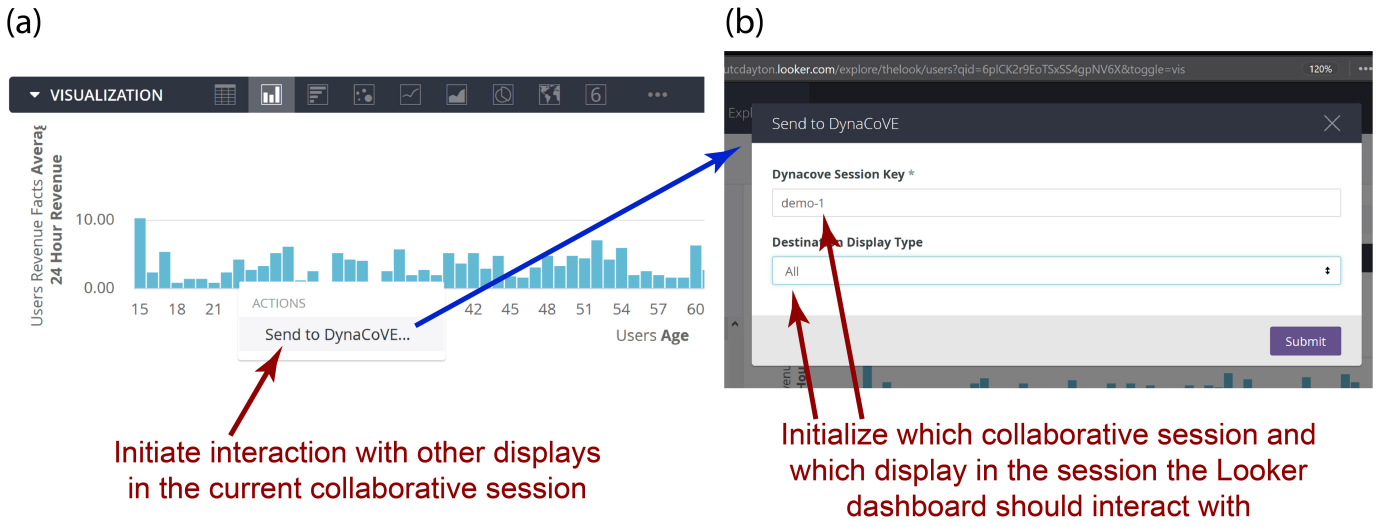


Fig. 4. Looker client use. (a) A context menu containing the DynaCoVE data action appears upon interacting with a dashboard. (b) A session ID is initialized, as multiple collaborative sessions may be running through the DynaCoVE server at any given moment.

on a data point, a message containing the data point and other data is sent to Looker's Action Hub server. A reusable *Send to DynaCoVE* Data Action was created and deployed a customized Action Hub which receives the Action and forwards its data to the DynaCoVE server. Once configured, the user interaction flow is as follows.

1. Build acompatible query against the data model, and select a visualization.
2. A user clicks on a visualization in a dashboard and is presented with a drop-down contex menu with a link to initiate the Send to DynaCoVE Data Action (Fig. 4a).
3. Optionally, a form can be presented with additional inputs (Fig. 4b). The Data Action is executed when the user clicks "Submit". If no form is defined, the Data Action is immediately executed.
4. A request is sent to the Action Hub, which forwards the data to the DynaCoVE server.
5. The DynaCoVE server broadcasts the data to all other clients.

4.2 DynaCoVE Server

DynaCoVE is arranged in a client-server architecture with the server acting as a "message broker" between clients. In this context a client is any system that is capable of displaying a visualization and receiving input either through a local GUI or from another client through the server. Different clients may provide distinct visualization, display, and user input capabilities, but care was taken to ensure these details have no bearing on the server in order to simplify future extension.

Collaborative visual analytics amongst clients operating at different geographic locations requires accessibility over the public internet. While a local install is one option, the main DynaCoVE server and Looker Action Hub are cloud hosted applications residing in AWS to enable the desired level of collaborative interaction. To protect the data in transit between Looker and the DynaCoVE Action Hub, the HTTP traffic is encrypted using 4096-bit TLS asymmetric key cryptographic certificates generated with the Let's Encrypt Certificate Authority and 4096-bit Diffie-Hellman key exchange parameters. Efforts are underway to provide equivalent security in the DynaCoVE server.

It is assumed that all clients that connect to the server are participating in a specific collaborative session. A uniqued session ID is needed to allow the server to properly route messages from one client to all other clients in the same session. Furthermore, to enable interaction all clients in a session are operating on datasets containing a common variable, which we call the shared key. This shared key must be configured when configuring a visual analytics session between clients operating on different data. It is left to the user to define a meaningful linkage between datasets. More details on the test data types is presented later. Fig. 5a illustrates a session with two clients connected, Looker for 2D interaction and VTK/VRUI for 3D interaction.

4.3 DynaCoVE Communication

Communications between the aforementioned modules was handled in a client-server model using the ZeroMQ distributed message passing library [26]. ZeroMQ supports many languages and many communication protocols, which is key given that one cannot predict which future components may need to be wrapped into DynaCoVE.

4.3.1 Scalability

Starting with a communication framework that will scale to large data was critical, as large transfers are anticipated as DynaCoVE is extended to encapsulate more tools. Further, some clients may eventually need to run in parallel so parallel communication support was desired. For these reasons, strong and weak scalability tests were run with ZeroMQ to verify that it would meet DynaCoVE's future needs. The scalability testing approach can be summarized as follows.

- A shell version of the DynaCoVE C++ client stub was modified to run in parallel with MPI.
- Weak (same size per core) and Strong (same total size) scaling tests were run on 16, 32, 64, 128 and 256 cores.
- Tests were run with vectors of doubles of length 2^5 up to 2^{18}.

Scalability results were promising. For a set amount of data per core, the execution time does not change significantly as more cores, and therefore more data, is added. For fixed data sizes, the data transfer time dropped near linearly in proportion to the number of cores. For this reason, ZeroMQ was selected as the underlying distributed message passing library used by DynaCoVE.

4.3.2 DynaCoVE Messaging Protocol

A DynaCoVE client-server communication protocol specification was defined based on ZeroMQ's Majordomo Protocol [27]. This is called the DynaCoVE Messaging Protocol. This specification defines an asynchronous two-way communication protocol that allows clients to send and receive control messages and arbitrary unstructured data.

Messages consist of a *message type* header followed by an optional *message body*. To serialize arbitrary data structures and eliminate differences in data representation between programming languages,

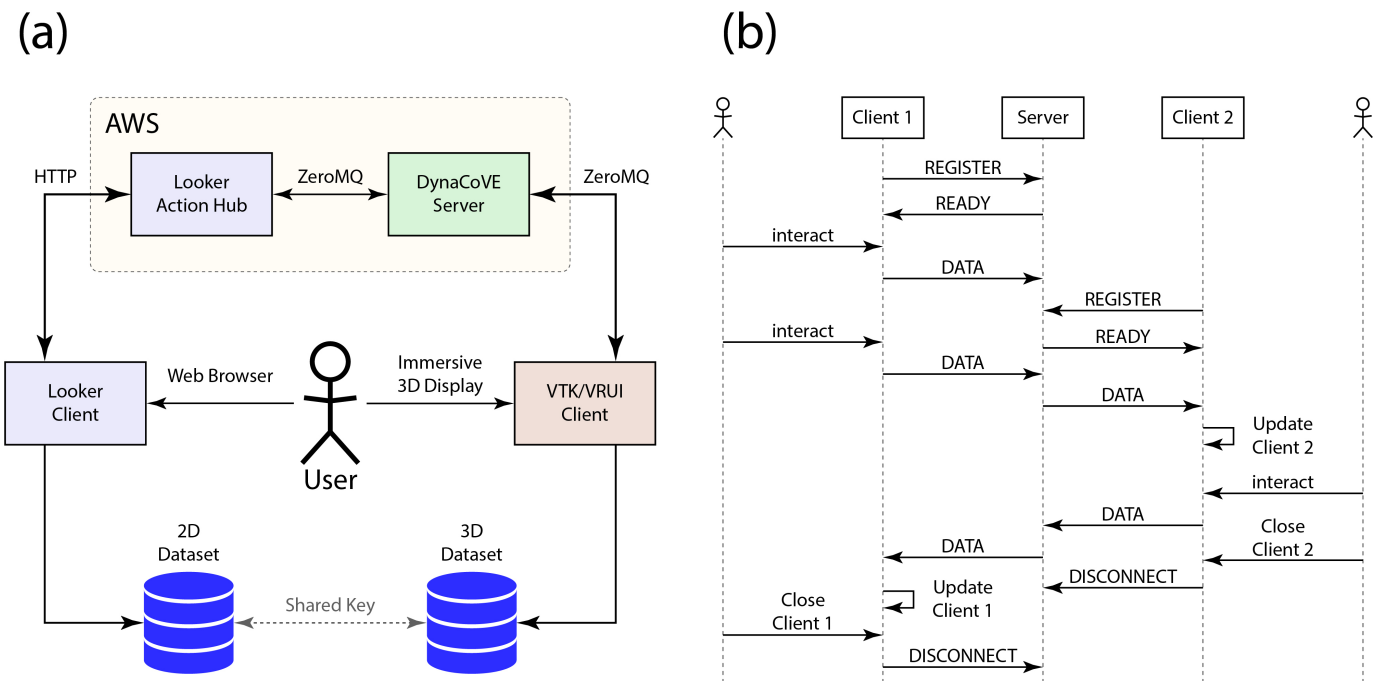


Fig. 5. Communication between components in dynacove. (a) Example collaborative visual analytics session between the DynaCoVE server and the Looker and VTK/VRUI clients. (b) Sequence diagram detailing messages exchanged during an example interactive visualization session involving two clients. HEARTBEAT messages are not shown.

message bodies are encoded in the popular JSON format [28]. Table 1 describes the message types defined by the protocol.

An example collaborative session involving two clients is illustrated in Fig. 5b. Interactions with both clients are capable of driving updates in the other clients as long as they are participating in the same session. This works the same way if the clients are running on the same computer or in different geographic locations on different types of displays.

4.3.3 DynaCoVE Interface

Clients in DynaCoVE typically have their own interface in order to setup visualizations. It is feasible to manually initialize each client, however when several clients are participating in a unified session it is desireable to be able to initialize them from a single interface. DynaCoVE provides a means to accomplish this called the meta-visualization graph (MVG). Inspired by the shader graph networks used in 3D modeling packages, the MVG is essentially a graph network that allows one to chain visualization algorithms together to define an output for each display in a session.

The MVG provides a means to specify datasets and the shared key, synchronize transient data when there is not an exact mapping between file numbers, and set up visualizations. A visualization interface module (VIM) is then used to parse the MVG, and produce client-specific configuration and initialization. When a new client is added to DynaCoVE, a new VIM must also be provided in order to take advantage of this capability. At this time, the MVG is specified in a text-based configuration file that is input during session setup, however work is currently underway to have a D3-based interface [29] capable of graphically building an MVG.

Message Type	Description
HEARTBEAT	Clients and the server continuously transmit HEARTBEAT messages with a reliable frequency to support detection of dropped or stale connections and prevent early termination.
ERROR	Error reporting. This message type is defined for future use, but is presently unutilized.
REGISTER	The first message transmitted by a client to the server, providing any data or metadata necessary to register the client.
READY	The server responds with a READY message after a client has been successfully registered.
DISCONNECT	A message indicating that the connection is about to be terminated.
DATA	A JSON-encoded arbitrary data message. Any data necessary to further route the message or handle its contents should be contained within the body of the message.

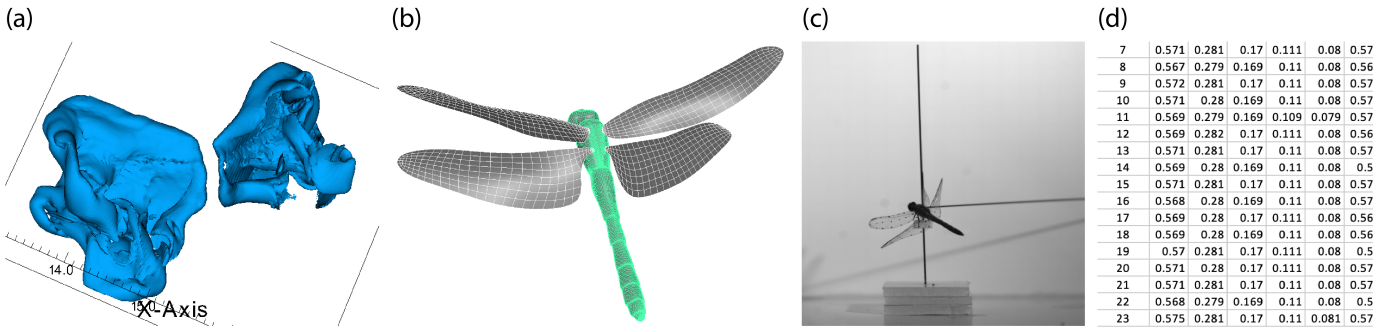
Table 1. DynaCoVE Messaging Protocol Message Types.

5 Results

The aforementioned visualization ecosystem components have been tested with a variety of data types and display hardware. The interaction updates are quite responsive. The only caveat here is that there is some lag when processing the whole unsteady vector field, as individual time steps are in excess of 600MB. However, this lab was no more severe than what is encountered in stand-alone visualization tools, so it is not believed to be related to some bottleneck in DynaCoVE's distributed communication. A record of the visual analtyics test cases run with DynaCoVE thus far is summarized in the remainder of this section.

5.1 Representative Test Data

To properly test DynaCoVE, one needs multiple related datasets with a variety of different spatial components and datatypes. The content of the datasets is actually irrelevant for our purposes, and could have been synthetically generated. However it is easier to evaluate DynaCoVE using datasets that the authors had a good feeling of how things ought to be looking in all of the visualizations. For this reasons, we used a series of datasets gathered from a previous study of dragonfly flight [30–32].



7	0.571	0.281	0.17	0.111	0.08	0.57
8	0.567	0.279	0.169	0.11	0.08	0.56
9	0.572	0.281	0.17	0.11	0.08	0.57
10	0.571	0.28	0.169	0.11	0.08	0.57
11	0.569	0.279	0.169	0.109	0.079	0.57
12	0.569	0.282	0.17	0.111	0.08	0.56
13	0.571	0.281	0.17	0.11	0.08	0.57
14	0.569	0.28	0.169	0.11	0.08	0.5
15	0.571	0.281	0.17	0.11	0.08	0.57
16	0.568	0.28	0.169	0.11	0.08	0.57
17	0.569	0.28	0.17	0.111	0.08	0.56
18	0.569	0.28	0.169	0.11	0.08	0.56
19	0.57	0.281	0.17	0.111	0.08	0.5
20	0.571	0.28	0.17	0.111	0.08	0.57
21	0.571	0.281	0.17	0.11	0.08	0.57
22	0.568	0.279	0.169	0.11	0.08	0.5
23	0.575	0.281	0.17	0.11	0.081	0.57

Fig. 6. Related but differently typed data sets used to test DynaCoVE. (a) Vector field data. (b) Mesh data. (c) Video data. (d) Tabular data.

The individual test datasets include the following.

- Unsteady 3D vector field data: 800 time steps of fluid simulation output in VTK format.
- High-speed camera data: Three videos with 80 time steps each from different perspectives taken at the same time in TIF format.
- 3D mesh data: 800 time steps of body and flapping wing surfaces previously extracted from video data in OBJ format.
- Tabular numeric database: 800 time steps of various measurements including lift and thrust coefficients and wing twist and camber measurements taken from both wings and the flow field at each time step.

All the datasets can be linked in a meaningful way with a shared key to enable interactive visualization due to the fact that they all share a common time scale. An additional challenge here was that while the physical time scale was the same, the number of discrete time steps available weren't equal for each dataset. For example, there are 800 time steps of vector field data that correspond to 80 time steps of high speed camera data. Any required scaling over the key variable is accounted for during session setup. The test data is shown in Fig 6.

5.2 Tested Data Combinations

The DynaCoVE clients for VisIt, VTK/VRUI, Looker were created in parallel with the server. While integrating the components and smoothing out unforseen issues, a series of increasingly complex text cases was employed. Table 2 denotes the exact configurations that have been run to date. These combinations were each test with several display options.

Tabular	Vector	Surface	Video
		VisIt	VisIt
	VisIt	VisIt	
Looker	VisIt	VisIt	VisIt
	VTK/VRUI	VisIt	
Looker	VTK/VRUI	VisIt	
Looker	VTK/VRUI	VisIt	VisIt

Table 2. Summary of cross display interactive visualizations that have been run with the DynaCoVE to date. The degree of difficulty increases top down.

5.3 Visualizations

With regard to the display technology, DynaCoVE was tested in Wright State University's Appenzeller Visualization Lab [33], which houses a variety of different 2D, 2.5D, and 3D display hardware for testing purposes. Specifically, the Appenzeller Lab currently includes a Barco Ispace, multiple 3D TVs with optical trackers, a passive projection screen with a Kinect sensor, a 48-megapixel tiled wall display, a Sony HMZ-T1 head-mounted display with optical trackers, and a custom built Display Infrastructure for Virtual Environments (DIVE) system with 3 walls and a 12 by 12 foot walkable footprint. Thus far, the majority of the testing has focused on the tiled wall display, the DIVE system with an attached head tracker, and on traditional 2D monitors.

The most basic first steps after integrating all of the DynaCoVE components was to mirror the same visualization in the same package on multiple computers. Updates initiated to the GUI of one display can then be reflected in all other displays in the session. Once this capability was in place several example collaborative visual analytics sessions were made on multiple computers with multiple display types running multiple visualization packages. An example of this is shown in Fig. 7. This figure illustrates the display agnostic characteristic of DynaCoVE, as the same client (VisIt, VisIt and Looker), data (mesh, video, and tabular) and visualization algorithm (wireframe, pseudocolor, and bar chart) combination is being run on two combinations of displays. This is a trivial change in the session setup.

A similar example is shown in Fig. **??**. Here the full DIVE system is running with the VTK/VRUI client. As long as it was working with the individual client software a priori, the 6-DOF head tracker inpute device works automatically with DynaCoVE in the 3D display with no special setup instructions.

It was discovered that there can be some synchronization issues when dealing with many displays in a collaborative visual analytics session, when one or more is operating on significantly larger datasets than the others. In our example cases, visualizations using the vector field data are significantly more cumbersome than all the other related datasets. The communication overhead imposed by the DynaCoVE server is negligible, so any lag imposed by data sizes is no different than what would be seen in individual packages on individual displays. However, the difference here is that one can continue interacting with a client that has a more lightweight dataset, unaware that they are queuing up updates in another display that is no longer in sync.

One way to resolve this issue is via a hierarchical data reduction scheme. Work is underway to incorporate the necessary data analytics clients into DynaCoVE to enable this option. Another option is to detect the situation on the server side and send cues to the respective clients to either reset the client that is behind or temporarilly prevent incoming instructions from other clients specifying additional interactions until the lagging client is once again up to date.

Tracking the situation on the server side can be done in real time. For example, in Fig. 8, a console log of interactions is shown in realtime as interactions propogate between VisIt and VTK clients. The server not only propogates user interactions, but it also tracks when actions are automatically completed elsewhere in response to user actions.

As mentioned earlier, DynaCoVE is intentionally display technology, and visualization algorithm agnostic. It is ultimately up to the user to define cross dataset linkages that will yield meaningfull visualizations, with DynaCoVE as the enabling framework. DynaCoVE's aim is to provide the necessary means to encapsulate existing tools in such a

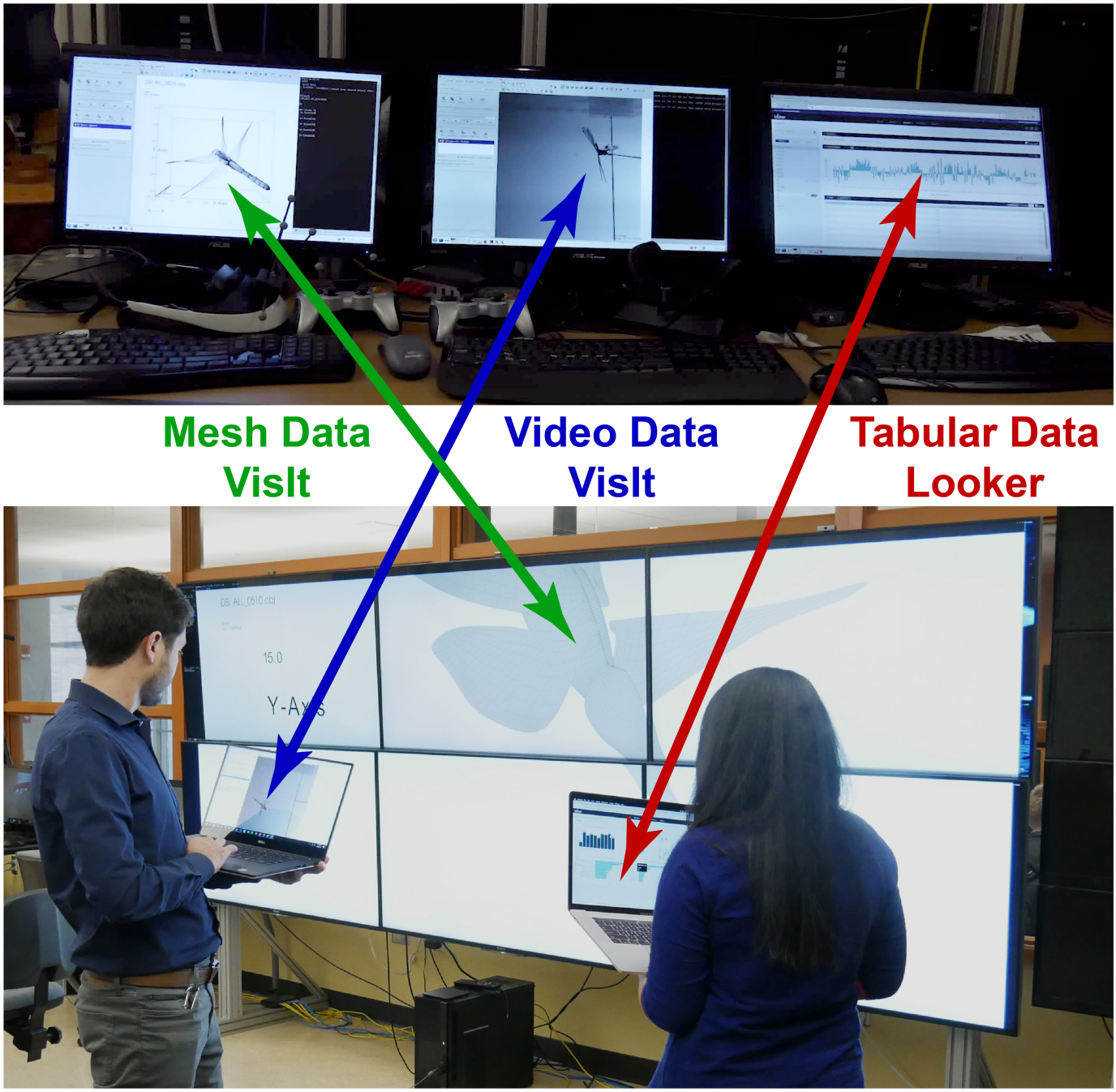


Fig. 7. Demonstration if display agnostic capabilities of DynaCoVE. Here, two collaborative visual analytics sessions are shown that were set up with the same combination of clients and datasets. Aside from ensuring that the client software is properly installed, there are no actual changes to DynaCoVE's configuration needed to operate in different display environments. Top: Three traditional monitors. Bottom: Two laptops and one tiled wall display.

way that cross-display, cross-package interactions can be efficiently performed.

6 Conclusion and Future Work

Based on the results achieved to date, DynaCoVE is believed to be successful first concrete implementation of a visualaization ecology [11]. In particular, the ability to have cross display interactions between disparate visualization capabilities was demonstrated, to an extent that is to the best of our knowledge, novel.

There are a few existing components of DynaCoVE that could be improved further. For example, the meta-visualization graph option to initialize many visualization packages from a common interface is still text based at this time and should be extended to a more user friendly graphical capability. Further, the visualization clients integrated into DynaCoVE currently only expose a subset of their capabilities to DynaCoVE, so this stands to be extended to make the tool more general purpose. Based on the lessons learned to date the future work on the DynaCoVE framework will focus on the following areas.

1. Integrate modern big data management components (Hadoop, Spark, etc.) into DynaCoVE.
2. Identify the potential users of DynaCoVE to better understand their needs. In particular, DynaCoVE is believed to be applicable to test and evaluation [34] as well as multidisciplinary engineer-

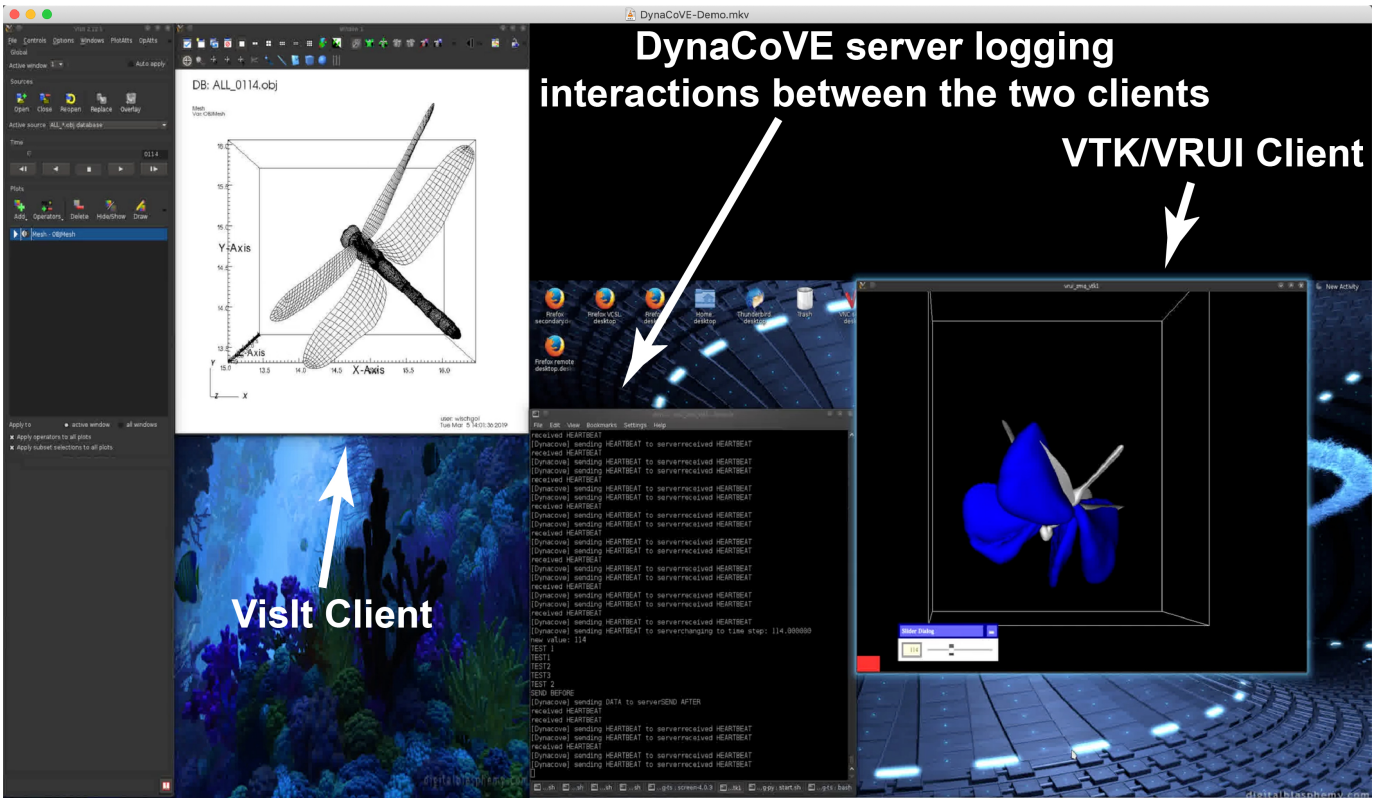


Fig. 8. Example of two VisIt clients (left and center displays) displaying mesh and video data for the dragonfly test dataset, and one Looker client with a bar chart with time on the x-axis and synthetic data on the y-axis. Each display is connected to a separate Linux workstation. Interactive updates occur in near realtime for these datasets.

ing.

3. Test DynaCoVE scalability for interactive visualizations that require large amounts of data transfer in near real time.
4. Provide additional sanity checks to ensure accurate visualizations. For example, identify when one display is out of sync with all other displays due to excessive queued up interactions, and automatically skip forward to align current states in all displays.
5. Extend DynaCoVE to include data analytics clients (data reduction, machine learning, clustering, etc.)
6. Extend DynaCoVE on a wider variety of display and input devices including non-visual systems like wearable sensors that can be monitored visually with DynaCoVE as well as 6-DOF and haptics input devices
7. Implement the full meta-visualization graph interface in a D3-based case setup client
8. Test DynaCoVE for collaborative visualization with users at different geographic locations

Acknowledgments

This work was supported in part by the Army Research Lab SBIR TODO a grant from XYZ (# 12345-67890).

References

[1] A. Endert, M. S. Hossain, N. Ramakrishnan, C. North, P. Fiaux, and C. Andrews, "The human is the loop: new directions for visual analytics," *Journal of Intelligent Information Systems*, vol. 43, pp. 411–435, December 2014.

[2] L. Zhang, A. Stoffel, M. Behrisch, S. Mittelstadt, T. Schreck, R. Pompl, S. Weber, H. Last, and D. Keim, "Visual analytics for the big data era — a comparative review of state-of-the-art commercial systems," in *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 173–182, October 2012.

[3] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler, *Visual Analytics: Scope and Challenges*, pp. 76–90. Springer Berlin Heidelberg, 2008.

[4] "Looker." `https://looker.com/`. Accessed: 2019-03-14.

[5] "Tableau." `https://www.tableau.com/`. Accessed: 2019-03-14.

[6] "Power bi." `https://powerbi.microsoft.com/en-us/`. Accessed: 2019-03-14.

[7] H. Doleisch, "Simvis: Interactive visual analysis of large and time-dependent 3d simulation data," in *Simulation Conference, 2007 Winter*, pp. 712–720, December 2007.

[8] K. Shi, H. Theisel, H. Hauser, T. Weinkauf, K. Matkovic, H.-C. Hege, and H.-P. Seidel, "Path line attributes - an information visualization approach to analyzing the dynamic behavior of 3d time-dependent flow fields," in *Topology-Based Methods in Visualization II*, Mathematics and Visualization, pp. 75–88, 2009.

[9] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser, "A four-level focus+context approach to interactive visual analysis of temporal features in large scientific data," *Comput. Graph. Forum*, pp. 775–782, May 2008.

[10] C. Koehler, R. Durscher, P. Beran, and N. Bhagat, "Adjoint-enhanced flow visualization," *Journal of Visualization*, vol. 21, pp. 819–834, October 2018.

[11] S. Su, M. Barton, M. An, V. Perry, C. Li, J. Jia, and B. Panneton, "Visual analytics ecology for complex system testing," in *Vis in Practice*, October 2017.

[12] S. Su, A. Chaudhary, P. O'Leary, B. Geveci, W. Sherman, H. Nieto, and L. Francisco-Revilla, "Virtual reality enabled scientific visualization workflow," in *2015 IEEE 1st Workshop on Everyday Virtual Reality (WEVR)*, pp. 29–32, March 2015.

[13] T. Marrinan, J. Aurisano, A. Nishimoto, K. Bharadwaj, V. Mateevitsi, L. Renambot, and L. Long, "Sage2: A new approach for data intensive collaboration using scalable resolution shared displays," in *10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 177–186, October 2014.

[14] D. Kobayashi, S. Su, L. Bravo, J. Leigh, and D. Shires, "Parasage: Scalable

web-based scientific visualization for ultra resolution display environment," in *IEEE Visualization, Poster*, October 2016.

[15] S. Su, V. Perry, N. Cantner, D. Kobayashi, and J. Leigh, "High-resolution interactive and collaborative data visualization framework for large-scale data analysis," in *2016 International Conference on Collaboration Technologies and Systems (CTS)*, pp. 275–280, October 2016.

[16] "Paraview web." `https://www.paraview.org/web/`. Accessed: 2019-03-14.

[17] R. Durscher and D. Reedy, "pycaps: A python interface to the computational aircraft prototype syntheses," in *AIAA Scitech Forum*, January 2019.

[18] R. Snyder, "Sensitivity analysis for multidisciplinary systems (sams)," tech. rep., Air Force Research Lab, January 2016.

[19] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[20] R. Niu, P. Zulch, M. Distasio, E. Blasch, G. Chen, D. Shen, Z. Wang, and J. Lu, "Joint sparsity based heterogeneous data-level fusion for multi-target discovery," in *2018 IEEE Aerospace Conference*, pp. 1–8, March 2018.

[21] K. Sohn, W. Shang, and H. Lee, "Improved multimodal deep learning with variation of information," in *Advances in Neural Information Processing Systems 27*, pp. 2141–2149, 2014.

[22] H. Childs, E. Brugger, B. Whitlock, J. S Meredith, S. Ahern, K. Bonnell, M. Miller, G. Weber, C. Harrison, D. Pugmire, T. Fogal, C. Garth, A. Sanderson, E. W. Bethel, M. Durant, D. Camp, J. Favre, O. Rübel, P. Navratil, and F. Vivodtzev, "Visit: An end-user tool for visualizing and analyzing very large data," *Proceed SciDAC*, pp. 1–16, January 2011.

[23] W. J. Schroeder, L. S. Avila, and W. Hoffman, "Visualizing with vtk: a tutorial," *IEEE Computer Graphics and Applications*, vol. 20, pp. 20–27, September 2000.

[24] "Vrui vr toolkit." `http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/`, 2015. Accessed: 2019-03-14.

[25] "Visit python interface manual." `http://visit.ilight.com/svn/visit/trunk/releases/2.10.0/VisItPythonManual.pdf`. Accessed: 2019-03-14.

[26] P. Hintjens, *ZeroMQ: messaging for many applications*. O'Reilly Media, Inc., 2013.

[27] "Zeromq majordomo protocol." `https://rfc.zeromq.org/spec:7/MDP/`. Accessed: 2019-03-14.

[28] "Javascript object notation (json)." `https://www.json.org/`. Accessed: 2019-03-14.

[29] M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-driven documents," *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[30] C. Koehler, T. Wischgoll, H. Dong, and Z. Gaston, "Vortex visualization in ultra low reynolds number insect flight," *IEEE TVCG*, vol. 17, pp. 2071–2079, Dec. 2011.

[31] C. Koehler, Z. Liang, Z. Gaston, H. Wan, and H. Dong, "3d reconstruction and analysis of wing deformation in free-flying dragonflies," *Journal of Experimental Biology*, vol. 215, no. 17, pp. 3018–3027, 2012.

[32] H. Dong, Z. Liang, H. Wan, C. Koehler, and Z. Gaston, "An integrated analysis of a dragonfly in free flight," in *28th AIAA Applied Aerodynamics Conference*, June 2010.

[33] T. Wischgoll, "Display systems for visualization and simulation in virtual environments," *Electronic Imaging*, no. 1, pp. 78–88, 2017.

[34] M. Barton and N. Raju, "Data-intensive computing for test and evaluation," *ITEA Journal*, vol. 38, no. 2, pp. 177–152, 2017.