SAND2019-0061C

VideoSwarm: Analyzing Video Ensembles

Shawn Martin, Milosz A. Sielicki, Jaxon Gittinger, Matthew Letter, Warren L. Hunt, Patricia J. Crossno
Sandia National Laboratories, Albuquerque, NM

Abstract

We present VideoSwarm*, a system for visualizing video ensembles generated by numerical simulations. VideoSwarm is a web application, where linked views of the ensemble each represent the data using a different level of abstraction. VideoSwarm uses multidimensional scaling to reveal relationships between a set of simulations relative to a single moment in time, and to show the evolution of video similarities over a span of time. VideoSwarm is a plug-in for Slycat, a web-based visualization framework which provides a web-server, database, and Python infrastructure. The Slycat framework provides support for managing multiple users, maintains access control, and requires only a Slycat supported commodity browser (such as Firefox, Chrome, or Safari).*

Introduction

Scientists and engineers are increasingly interested in analyzing ensembles of results generated by numerical simulations [22]. These ensembles are produced by varying parameters of the model for different runs of the simulation. Ensemble analysis has been used with climate data [28, 30] and computational fluid dynamics models [13]. Ensembles can be used to understand the uncertainty in model parameters [10], the behavior of the model in relation to the physical system [35], and ultimately the properties of the physical system itself.

Often, however, ensembles can contain cumbersome amounts of data. The simulations are usually three-dimensional (requiring grid points at a suitable spatial resolution); they often produce multivariate data (e.g. pressure, velocity, temperature, etc.) at each grid point; and finally the size of this data is multiplied by the number of time steps. A single simulation run can yield gigabytes to potentially terabytes of data. Further, large ensembles might contain thousands of simulations. Therefore, practical concerns often require the *in situ*[1] computation of features, statistics, or location-based samples to analyze the results [22]. Sometimes storage constraints dictate that only the calculated features can be retained after running the simulations.

In this paper, we use *in situ* produced videos rendered from multiple viewpoints as the simulation features. Videos are commonly used to understand simulation results and have the added advantage of being generic for any type of simulation. Videos have a smaller storage requirement than full simulation results, and may contain more information than summary statistics or location-based samples.

Given a set of simulation videos, several questions arise: How can we compare and contrast videos in our ensemble? Where in time are the videos similar and where do they diverge? Do the videos cluster? Can we identify interesting behavior in the videos and when it occurs? In other words, without actually watching every video, how can we digest the full content of the data? To answer these questions, we have developed a tool for analyzing simulation video ensembles, called *VideoSwarm*.

[1]Running on the high performance computer (HPC) concurrently or in lock step with the simulation.

VideoSwarm is a light-weight interface for interactive visualization of simulation video data, including the ability to view selected videos concurrently. VideoSwarm visualizes the video ensemble with two linked representations, each providing a different abstraction of the video data. In the first, multidimensional scaling [3] is used to provide a two-dimensional map of the relationships between the movies at a single user selected time within the simulation. In the second, the preceding maps for individual time step are indexed by one-dimensional video trajectories, which are also computed using multidimensional scaling. The combined effect of these representations is that the maps give the user a series of snapshots showing how the videos are related to each other, while the trajectories give the user a sense of how the videos progress over time and when interesting frames occur in the videos. The maps and trajectories also function as control mechanisms for selecting individual videos and time instances within the videos for synchronized viewing.

VideoSwarm is implemented as a plugin for Slycat [6], a system which provides a web server, a database, and a Python infrastructure for remote computation (on the web server). The Slycat VideoSwarm plugin is a web application which provides the previously described video ensemble analysis capability. There is no installation and the only requirement is the presence of a Slycat supported browser (e.g. Firefox). In addition, VideoSwarm supports (via Slycat) managment of multiple users, multiple datasets, and access control, therefore encouraging collaboration while maintaining data privacy. Slycat and VideoSwarm are implemented using HTML5, JavaScript, and Python. Slycat is open source (github.com/sandialabs/slycat). It is anticipated that an open-source version of VideoSwarm will be released in the future as part of Slycat.

In this paper, we describe VideoSwarm in detail. We discuss: our motivation and design goals; related work; the algorithms we use to represent the video ensembles; and the VideoSwarm user interface. We demonstrate our system using a video ensemble dataset obtained from a numerical simulation of a punch impacting a metal plate.

Although we use specific algorithms in this paper, it is important to note that the user interface is completely decoupled from the chosen algorithms. Alternative or new algorithms can be easily substituted for the algorithms described.

Motivation and Design Goals

In our earlier work on Slycat, we developed approaches for remotely retrieving and playing video ensemble data within the

Parameter Space model [7]. Configurable axes within the Parameter Space scatterplot, interactive color-coding of runs by variable values, and filtering features assisted with down selecting the number of videos for examination. Synchronized playback enabled direct comparisons between a handful of videos, which could be corresponding videos from different runs, or videos rendered from different viewpoints within the same run. Although this functionality provided a starting point for analyzing video results, it lacks the ability to scale as the number of runs increases.

We realized that Slycat needed an analysis technique that provides an understanding of video content at a higher level of abstraction. Akin to the way that text analysis provides an overview for the content found in a set of documents, we want to represent overviews of the content for a collection of videos. This led us to develop the following set of tasks that a video analysis model should facilitate:

(T1) Understand ensemble results without viewing all videos.

(T2) Cluster videos both in terms of frame similarity and in terms of temporal evolution.

(T3) Correlate video clusters with simulation input parameter values.

(T4) Evaluate sensitivity of video results to input parameters.

(T5) Find video outliers or videos demonstrating anomalous behavior over time.

(T6) Compare small sets of videos during playback.

We will use these demarcations (T1-T6) in the text that follows to show how we address these goals.

Related Work

Our work falls within the realm of ensemble visualization [22], a field which overlaps with uncertainty visualization, computational steering, and parameter space exploration.

The field of ensemble visualization is not entirely well-defined. Obermaier *et al.* [21] state that "being able to identify contributions of individual runs while at the same time providing summary statistics is one of the key capabilities of ensemble visualization." Some examples of ensemble visualization include isocontour "spaghetti" plots to visualize weather data [30]; isosurfaces for visualizing ensembles of three-dimensional data [2]; means and variances of climate simulation quantities [27, 28]; contour box-plots [36]; topological approaches [23,24]; and comparative visual analysis of two-dimensional ensembles [26].

In contrast to ensemble visualization, uncertainty visualization is concerned mainly with the statistical quality of the ensemble data, and in particular how it is represented visually [9,11,14,25]. Some examples of uncertainty visualization that are related to ensemble visualization are two-dimensional vector field glyphs that represent uncertainty in direction and magnitute [37], flow behavior of vector fields [12], and parameter sensitivity [10].

Computational steering refers to user guided optimization. For simulation ensembles, a feedback loop is implemented where results are visualized and the user changes input parameters to produce a desired outcome [15, 20]. Computational steering with ensembles has been used to good effect in tool design [5, 19].

Parameter space exploration is similar to computational steering, but the visualization is outside the optimization, or the optimization is absent [17,31]. Nevertheless, parameter space exploration generally focuses on identifying optimal parameters, or regions of optimal parameters [18, 29]. Parameter space exploration has been used for both image analysis [29] and video analysis [4].

The main difference between our work and previous work is that we are particularly interested in user interaction and relating all abstractions directly to the original data. Like the methods described in the field of ensemble visualization, we produce visualizations designed to abstract the entire ensemble while identifying contributions of individual runs. However, our abstractions are unlike previous work in that our visualizations are dynamic and change as the user examines different time points in the ensemble. Like parameter space exploration, we seek to identify interesting simulation inputs. However, we are not seeking particular parameters or regimes, but are seeking a general understanding of the data, such as the identification of clusters of similar videos and correlations of clusters with metadata.

In terms of subject matter, our work most closely relates to the parameter space exploration work of Bruckner and Möller [4], though they worked with volumetric data, while we are limited to making our comparisons based on image data. Although both systems are evaluating the impact of parameter changes on video sequences made by simulations, *in situ* workflows mean that we never have access to the intermediate volume data. In terms of algorithms, our trajectories are similar to the ensemble traces described in the work by Singh *et al.* [32]. The main difference between our trajectories and the ensemble traces being that we compute two-dimensional trajectories individually then align them, while all coordinates are computed simultaneously for the three-dimensional ensemble traces.

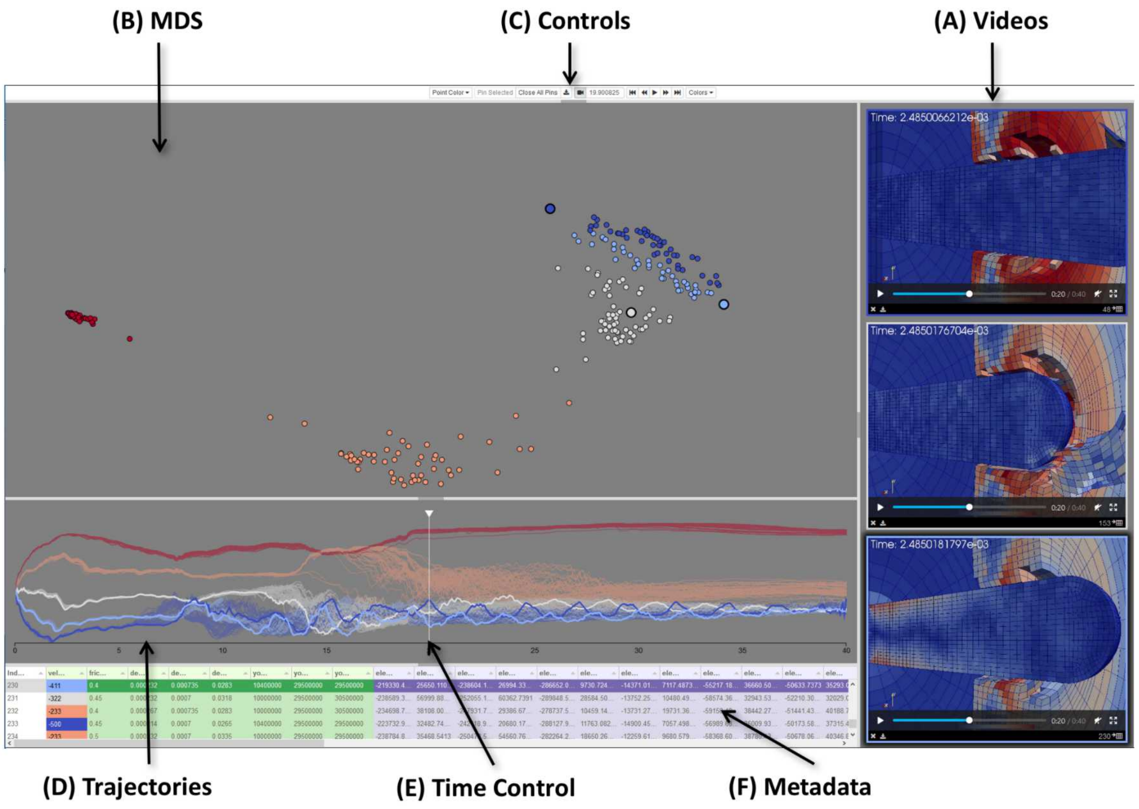
Algorithms

VideoSwarm provides an interactive user interface which organizes ensemble video data for easier comparison, contrast and identification of interesting behavior. The interface is based on two abstractions. The first abstraction provides a visualization of the videos at a given time point using dimension reduction via multidimensional scaling, and the second provides visualizations of the videos as time series plots, which we call *trajectories*.

Although we describe the algorithms used to compute our abstractions below, we again note that these particular algorithms are not required. The computations in practice amount to preprocessing steps which produce the data necessary to use the VideoSwarm interface. Any of a number of different distance matrices and/or algorithms could be easily substituted for those chosen below.

Multidimensional Scaling

VideoSwarm uses classical multidimensional scaling (MDS) [3] to provide a two-dimensional visualization of the videos in an ensemble. This visualization is displayed in the main frame of the VideoSwarm user interface, as shown in Figure 1(B). To be precise, suppose we have a dataset $\{v_i\}$, where v_i is a video in our ensemble. Since our videos originate from a numerical simulation, we assume that they all have the same number of frames, and that each frame is the same size. We then denote frame t of



Ind...	vel...	fric...	de...	de...	de...	yo...	yo...	yo...	ele...	ele...	ele...	ele...	ele...	ele...	ele...	ele...	ele...	ele...	ele...	ele...
230	-411	0.4	0.000[illegible]32	0.000735	0.0283	10400000	29500000	29500000	-219330.4...	25650.110...	-238604.1...	26994.33...	-286652.0...	9730.724...	-14371.01...	7117.4873...	-55217.18...	36660.50...	-50633.7373	35293.[illegible]
231	-322	0.45	0.000[illegible]32	0.0007	0.0318	10000000	29500000	30500000	-238589.3...	56999.88...	[illegible]52055.1...	60362.7391	-289848.5...	28584.50...	-13752.25...	10480.49...	-58574.36...	32943.53...	-52210.30...	32029.[illegible]
232	-233	0.4	0.000[illegible]67	0.000735	0.0283	10000000	29500000	30500000	-234698.7...	38108.00...	-2[illegible]7931.7...	29386.67...	-278737.5...	10459.14...	-13731.27...	19731.36...	-5915[illegible]	38442.27...	-51441.43...	40188.7
233	-500	0.45	0.000[illegible]14	0.0007	0.0265	10400000	29500000	29500000	-223732.9...	32482.74...	-242[illegible]18.9...	20680.17...	-288127.9...	11763.082...	-14900.45...	7057.498...	-56989.[illegible]	[illegible]6009.93...	-50173.58...	37315.4
234	-233	0.5	0.000[illegible]32	0.0007	0.0335	10000000	29500000	29500000	-238784.8...	35468.5413	-25047[illegible].5...	54560.76...	-282264.2...	18650.26...	-12259.61...	9680.579...	-58368.60...	38780[illegible]	-50678.06...	40346.8

Figure 1: VideoSwarm User Interface. Here we show VideoSwarm running in Firefox on a Windows desktop: (A) simulation video data selected by the user for playback/examination; (B) MDS coordinates of the videos in the ensemble, where each point corresponds to a particular video and the coordinates change when the user changes the time in the trajectories pane; (C) general controls allowing selection of metadata, colormaps, and video playback options; (D) video trajectories showing the progress of each video over its lifetime; (E) time selection controls allow the user to navigate the ensemble in time, with corresponding updates to the MDS coordinates and video displays; and (F) metadata table showing information available for each video in the ensemble, including physical parameters relevant to the analysis.

video v_i as a vector $\mathbf{f}_{it} = [f_{itk}]$, where f_{itk} is the kth pixel in the frame. Note that for color images we will have $3 \times m \times n$ pixels when an image is of size $m \times n$.

For each time point, we compute a pairwise distance matrix

$$D_t = \begin{bmatrix} d(\mathbf{f}_{1t}, \mathbf{f}_{1t}) & d(\mathbf{f}_{1t}, \mathbf{f}_{2t}) & \cdots \\ d(\mathbf{f}_{2t}, \mathbf{f}_{1t}) & d(\mathbf{f}_{2t}, \mathbf{f}_{2t}) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}, \tag{1}$$

where $d(\mathbf{f}_{it}, \mathbf{f}_{jt})$ gives a distance between video i and video j at frame t. For example using Euclidean distance we would have

$$d(\mathbf{f}_{it}, \mathbf{f}_{jt}) = \sqrt{\sum_k (f_{itk} - f_{jtk})^2}. \tag{2}$$

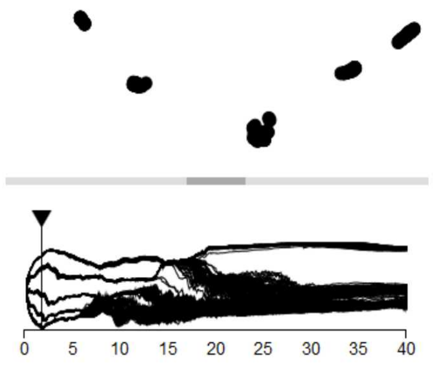
Other distances can be used, so that each distance metric can be tailored to the videos under analysis.

Now we use the MDS algorithm to compute coordinates for each distance matrix D_t. Since the MDS algorithm works on any distance matrix, we simplify notation by using D to represent D_t. The first step in MDS is to double center the distance matrix, obtaining

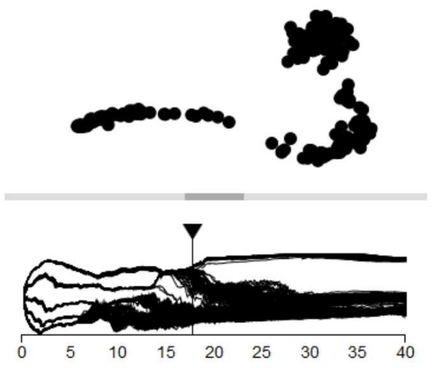
$$B = -\frac{1}{2} H D^2 H, \tag{3}$$

where D^2 is the componentwise square of D, and $H = I - \mathbf{1}\mathbf{1}^T/n$, n being the size of D. Next, we perform an eigenvalue decomposition of B, keeping only the two largest positive eigenvalues λ_1, λ_2 and corresponding eigenvectors $\mathbf{e}_1, \mathbf{e}_2$. The MDS coordinates are given by the columns of $E\Lambda^{1/2}$, where E is the matrix containing the two eigenvectors $\mathbf{e}_1, \mathbf{e}_2$ and Λ is the diagonal matrix containing the two eigvenvalues λ_1, λ_2. Resuming our use of the parameter t, we denote by E_t the two-column matrix containing the MDS coordinates for time t.

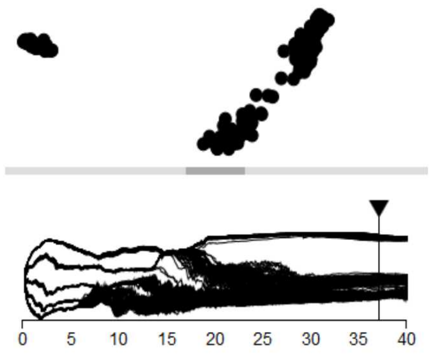
After MDS, we have two-dimensional coordinates representing the video ensemble for each time point. Watching these representations over time allows the VideoSwarm user to see how the videos are related, and how they associate and disassociate as the simulation progresses. However, it is important to note that the eigenvectors computed by MDS are unique only up to sign. This



(a) Time Control Near Video Start



(b) Time Control Near Video Middle



(c) Time Control Near Video End

Figure 2: Time Control. The VideoSwarm time control is a triangular slider that moves horizontally above the video trajectories. The time control is connected to a vertical line which intersects the trajectories at the current selected time. As the user drags the time control, the MDS coordinates representing the video ensemble at the selected time are updated, showing in real-time changes in the relationships between videos in the ensemble. In this figure we show (A) the time control just past the beginning of the videos in the ensemble, (B) near the midpoint of the videos in the ensemble, and (C) towards the end of the videos .

fact can manifest itself as disconcerting directional flips in the coordinates given even small changes in time by the analyst. To minimize these flips, we use the Kabsch algorithm [16] to compute an optimal transformation so that as we advance in time, each set of coordinates are as closely aligned to the coordinates from the next time step as possible.

The Kabsch algorithm uses the Singular Value Decomposition (SVD) to compute an optimal orthogonal transformation (rotations and reflections). If we assume that matrices P and Q have columns containing the consecutive time step MDS coordinates, then we form $A = P^T Q$ and use the SVD to obtain $A = U\Sigma V^T$. The transformation matrix is given by

$$T = VU^T. \tag{4}$$

We note a few important subtleties here. First, the Kabsch algorithm in it's entirety does not allow for reflections, only rotations. We do not implement this restriction and allow reflections as well as rotations. Second, we have found that restricting ourselves to two MDS dimensions prior to the coordinate alignment by the Kabsch algorithm can result in jumpy transitions between frames. For this reason, we allow the user to specify how many dimensions to keep for the frame alignment. This results in smoother from alignment when, for example, the MDS eigenvalues are similar in magnitude. After the frames are all aligned, projection back down to two dimensions is performed for the final visualization. Finally, we perform the frame alignment going backwards in time starting with the last frame, since the initial frame is typically identical for numerical simulations.

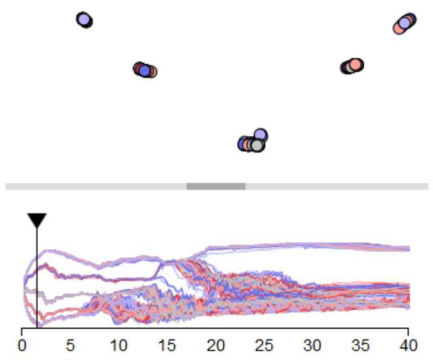
Trajectories

To efficiently index the two-dimensional MDS coordinate maps generated in the previous section, we also calculate time series trajectories for each video in the ensemble. These trajectories are displayed using the first of the two MDS coordinates described in the previous section. The trajectories are plotted below the main frame in the VideoSwarm interface, as shown in Figure 1(D).

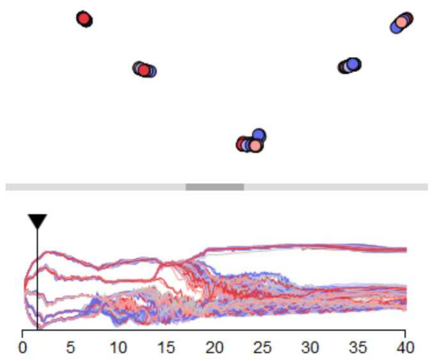
Recall we used E_t to denote the two-column matrix containing the Kabsch corrected MDS coordinates for the video ensemble at time t. If we write $E_t = [x_{it}, y_{it}]$, where (x_{it}, y_{it}) gives the (x,y) coordinates of video i at time t, then our trajectories are given by $z_i(t) = x_{it}$, plotted as a traditional time series.

Computational Costs

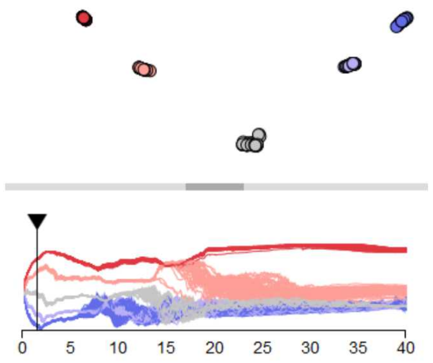
VideoSwarm depends on computing a sequence of pairwise distance matrices and a corresponding sequence of MDS coordinates. A single distance matrix calculation is $O(n^2 m)$ when n is the number of videos and m is the number of pixels in a video



(a) Coloring by Punch/Plate Friction



(b) Coloring by Plate Density



(c) Coloring by Punch Velocity

Figure 3: Coloring by Metadata. VideoSwarm allows the user to color both points in the MDS scatter plot and the video trajectories using metadata imported with the ensemble. On the left (A), we show our ensemble colored by punch/plate friction; in the middle (B), it is colored by plate density; and on the right (C), by punch velocity. From the colorings, it is clear that punch velocity is driving the formation of the clusters in the ensemble.

frame. We must compute the matrix for every frame, which is therefore $O(n^2mf)$ when f is the number of frames.

For each distance matrix, we then compute MDS coordinates using that distance matrix. The computational complexity of MDS is $O(n^3)$, so the cost of computing the sequence of MDS coordinates is $O(n^3f)$. In addition, it is worth noting that for large video ensembles (large n), there are faster methods for computing MDS. These methods vary in terms of speedup over classical MDS, as well as accuracy [1, 8, 38]. It is also worth noting that both the distance matrix and MDS coordinate calculations are embarassingly parallel in terms of the video frames.

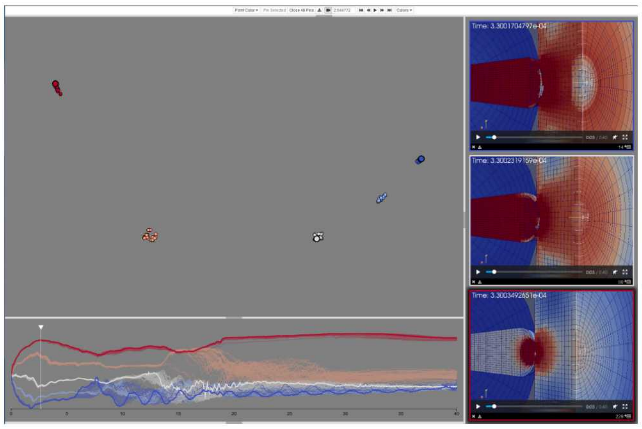
In our experiments, we examined four datasets using VideoSwarm (one of which is described in this paper). The processing time using a single thread on a laptop varied between five minutes and 8 hours. Our first dataset had a small number of short, low-resolution videos (256 videos, 50 frames per video, and 400 $\times$ 800 pixels per frame), and took approximately five minute to process. Our second dataset had numerous short, low-resolution videos (1441 videos, 90 frames per video, and 440 $\times$ 304 pixels per frame), and took approximately 20 minutes to process. Our third dataset had a small number of short, high-resolution videos (200 videos, 96 frames per video, and 1920 $\times$ 1080 pixels per frame), and took approximately one hour to process. Our final dataset (used as our example in this paper), had a moderate number of long, high-resolution videos (250 videos, 1001 frames per video, and 1024 $\times$ 768 pixels per frame), and took 8 hours. None of the datasets were processed using high performance/parallel computers.

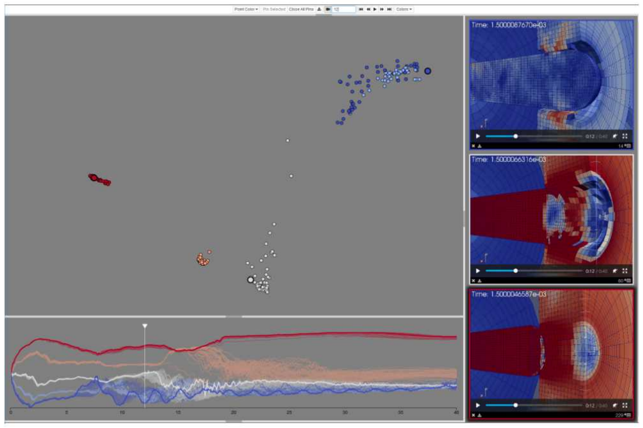
User Interface

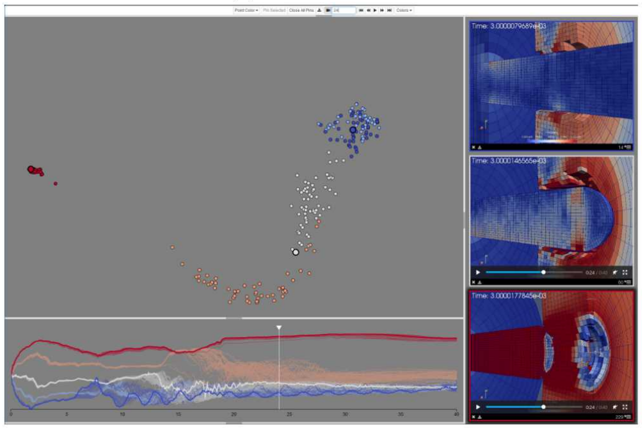
The VideoSwarm user interface provides interactive access to the analysis techniques described in the Algorithms section. Both the MDS coordinates and video trajectories are precomputed for maximum flexiblitiy in terms of video ensemble distance metrics, dimension reduction algorithms, and calculation of trajectories. Pre-computing the MDS and trajectory representations of the ensemble also ensures that VideoSwarm will provide real-time interactive exploration for ensembles.

The VideoSwarm interface consists of several panes, all of which are interactive and linked. The panes allow browsing the MDS projection, the trajectories, and the metadata, as well as viewing the videos themselves. The interface is shown in Figure 1.

The VideoSwarm interface is designed to help an analyst interactively investigate a video ensemble. The linked panes are used to display different levels of generality. The trajectories provide the level of greatest generality, displaying the ensemble as a whole over its entire play time in one two-dimensional plot. The MDS scatter plot provides a lower level of generality, showing how the videos are related on a frame by frame basis. Finally, the


(a) Videos Near Start


(b) Videos At 12 Seconds


(c) Videos At 24 Seconds

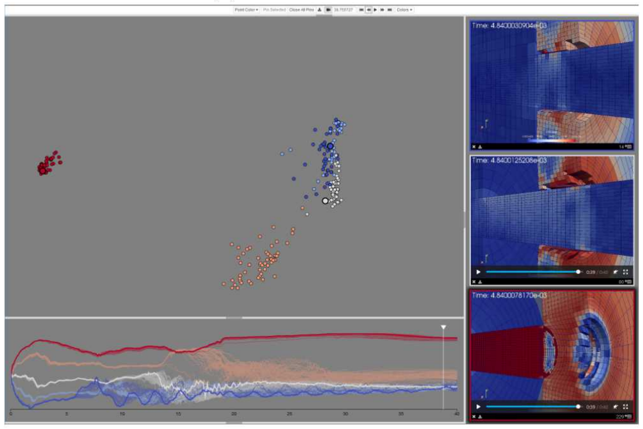

(d) Videos Near End

Figure 4: Synchronized Video Playback. Videos selected using the MDS scatterplot or the time series trajectories are shown in the video pane for viewing. These videos can be played independently or synchronized with the VideoSwarm time control. In this figure, we show synchronized playback of the videos selected from the red, white, and blue clusters (arranged in the video pane from top to bottom). We show the videos near their start (A); the videos at 12 seconds (B); at 24 seconds (C); and the videos near their end (D).

video pane and the metadata table allow the user to examine the actual simulation inputs/outputs and corresponding videos. All panes are linked by time.

The user can select an arbitrary number of videos and the video pane can be resized to show a larger number of small videos or a smaller number of large videos. If too many videos are selected to display at once, the user can scroll the video pane. In addition, videos can be played full screen if desired.

VideoSwarm is written in HTML5 and JavaScript using jQuery for the controls. The trajectories and MDS plots are rendered and animated using D3 and canvas, and the metadata table uses SlickGrid, an open source JavaScript component.

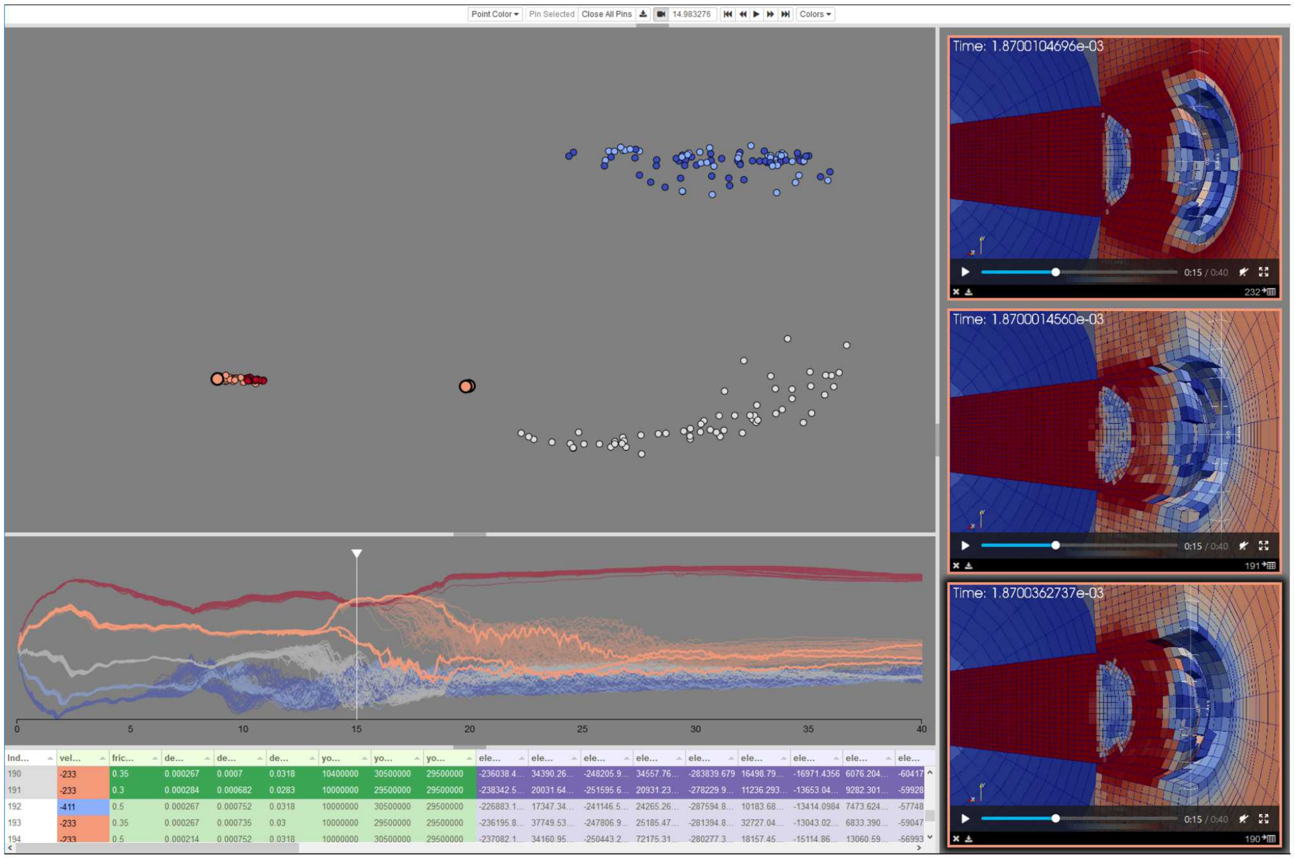
Example

We demonstrate the operation of VideoSwarm using an ensemble generated with Sierra/SolidMechanics (Sierra/SM) [33], a Lagrangian, three-dimensional code for problems with large deformations and nonlinear material behaviors. This ensemble was created to explore the effects of changes in simulation parameters on material fracturing. The modeled object is a punch impacting a metal plate under various conditions, such as different punch velocities, material properties, or plate thicknesses. For each run, the ensemble consists of 8 inputs and 38 outputs (12 scalar results, 6 event-triggered images, 16 variables changing over time, and 4 videos). The full ensemble is 15K runs, with about a terabyte of data (including a total of 90K images and 60K videos). We have randomly sampled that ensemble to create a smaller ensemble of 250 runs with videos created from three different viewpoints. Each video has a resolution of 1024 $\times$ 768, and is 40 seconds long with 1000 frames.

As mentioned in the User Interface section, and shown in Figure 1(B), VideoSwarm provides a central pane containing a scatter plot populated with the two-dimensional MDS coordinates. Each point in the scatter plot represents a particular video at a particular time. The relative positions of the points show how the videos are related at that time.

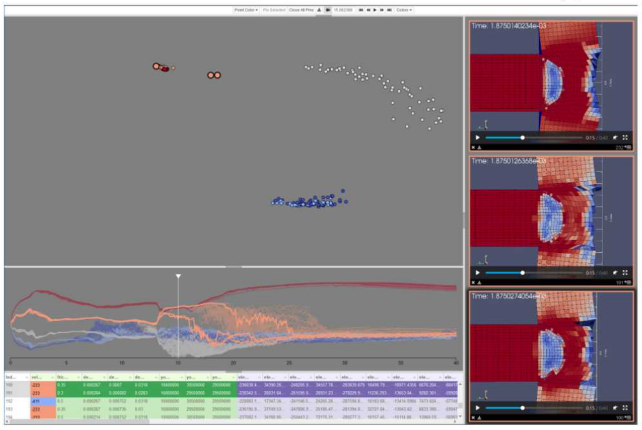
Below the scatter plot is a pane showing the video trajectories, seen in Figure 1(D). Each video trajectory is a standard time series plot. Like the points in the MDS scatter plot, a single time series plot represents a single video. Unlike the points in the scatter plot, a time series plot represents a video in its entirety (over its full time span).

Within the trajectories pane there is a triangular control connected to a vertical line indicating the current time in the ensemble. This is the time control shown in Figure 1(E). The time control is connected to the central MDS scatter plot so that drag-

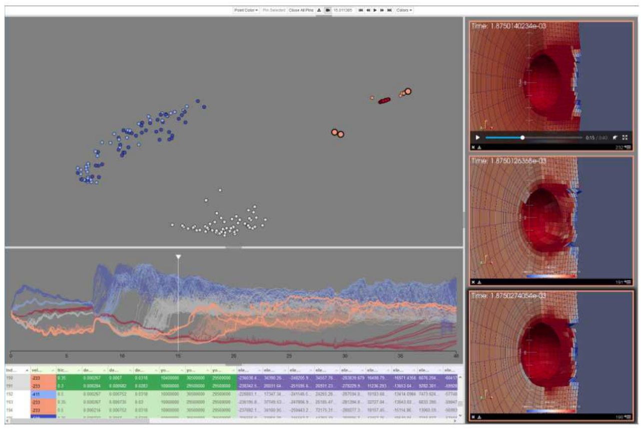


Ind...	vel...	fric...	de...	de...	de...	yo...	yo...	yo...	ele...	ele...	ele...	ele...	ele...	ele...	ele...	ele...	ele...
190	-233	0.35	0.000267	0.0007	0.0318	10400000	30500000	29500000	-236038.4...	34390.26...	-248205.9...	34557.76...	-283839.679	16498.79...	-16971.4356	6076.204...	-60417
191	-233	0.3	0.000284	0.000682	0.0283	10000000	29500000	29500000	-238342.5...	20031.64...	-251595.6...	20931.23...	-278229.9...	11236.293...	-13653.04...	9282.301...	-59928
192	-411	0.5	0.000267	0.000752	0.0318	10000000	30500000	29500000	-226883.1...	17347.34...	-241146.5...	24265.26...	-287594.8...	10183.68...	-13414.0984	7473.624...	-57748
193	-233	0.35	0.000267	0.000735	0.03	10000000	29500000	29500000	-236195.8...	37749.53...	-247806.9...	25185.47...	-281394.8...	32727.04...	-13043.02...	6833.390...	-59047
194	-233	0.5	0.000214	0.000752	0.0318	10000000	30500000	29500000	-237082.1...	34160.95...	-250443.2...	72175.31...	-280277.3...	18157.45...	-15114.86...	13060.59...	-56993

(a) Bottom Side View



(b) Side View



(c) Plate Top Side View

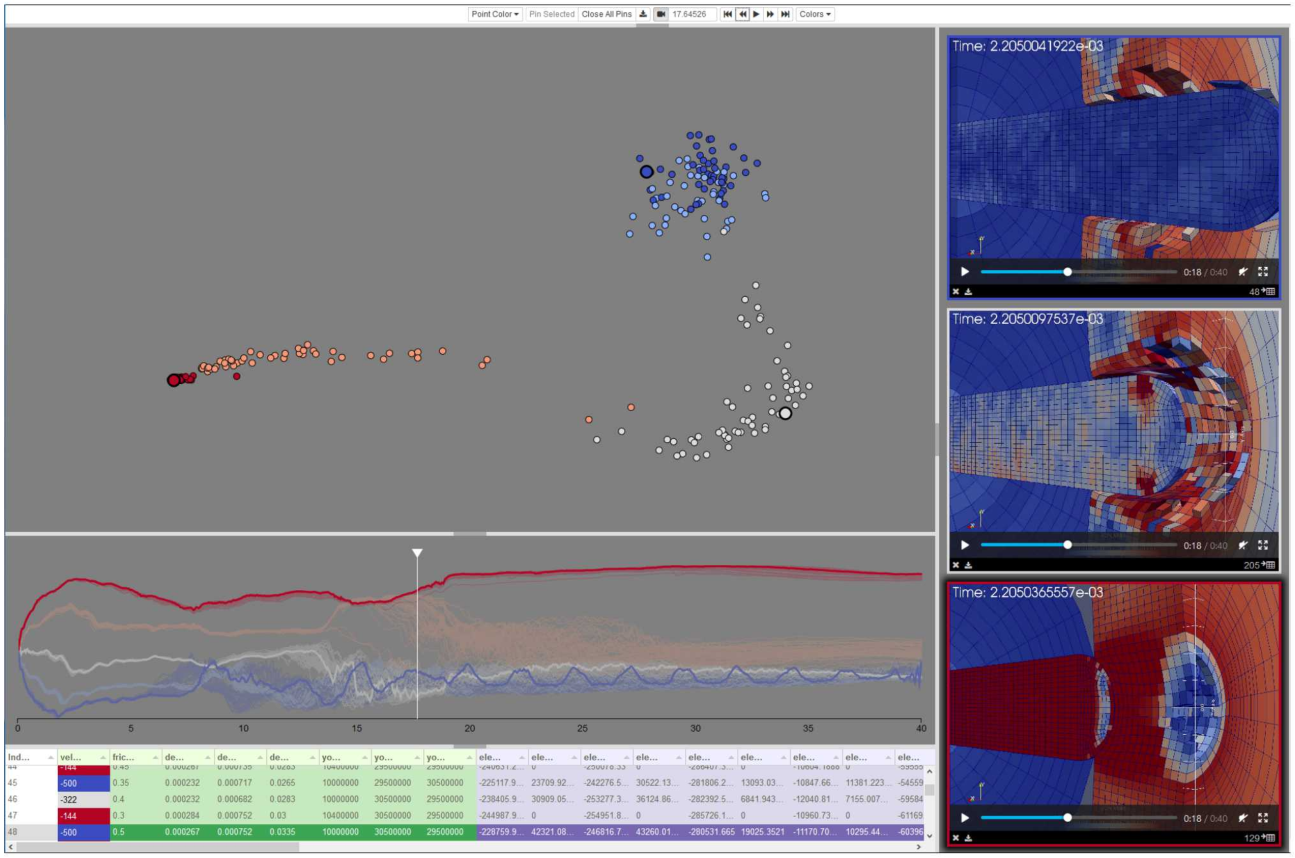
Figure 5: Anomalous Runs. Here we compare two anomalous runs with a run in the main cluster. We show videos (A) generated from a bottom side view; (B) from the side; and (C) of the plate only (punch not drawn) from a top side view. In all cases, the anomalous runs have lower stress in the plate (they are less red).

ging the time control updates the positions of the points in the MDS scatter plot corresponding to the changes in similarity of the videos at the current time.

Using the time control the user can see how the ensemble changes forwards and backwards in time. This enables the user to accomplish the clustering task (T2). Clusters are observed to form, merge, and re-coalese by manipulating the time control, as shown in Figure 2. The name VideoSwarm was inspired by the swarm-like behavior that the MDS scatter plot points exhibit as the time control is moved.

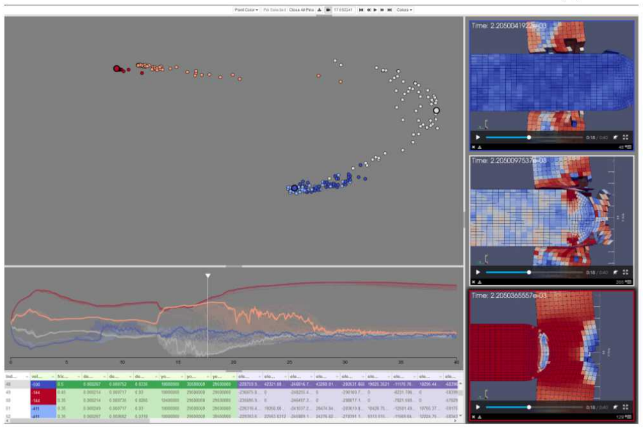
Metadata coloring plays a central role in the MDS and trajectory panes, and is used to support design tasks (T3) and (T4): examining correlations between video clusters and simulation metadata. Metadata coloring allows the user to select any column in the metadata table in Figure 1(F) and use the the corresponding value to color both the MDS representations and the video trajectories. Different metadata colorings are shown in Figure 3.

In the punch-plate example, the metadata coloring can be

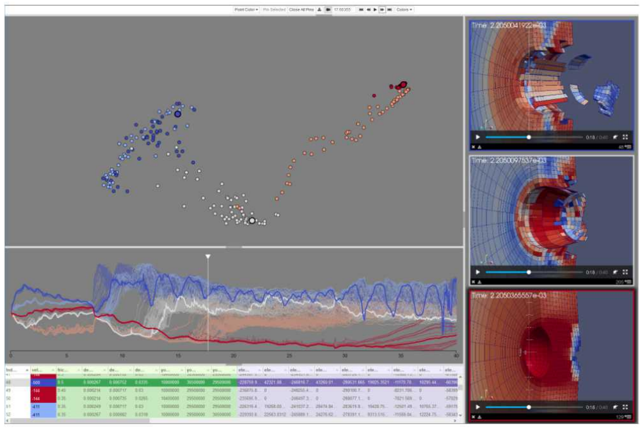


Ind...	vel...	fric...	de...	de...	de...	yo...	yo...	yo...	ele...	ele...	ele...	ele...	ele...	ele...	ele...	ele...	ele...
44	-144	0.45	0.000267	0.000735	0.0283	10400000	29500000	29500000	-240631.2...	0	-250078.33	0	-280407.3...	0	-10604.1888	0	-59555
45	-500	0.35	0.000232	0.000717	0.0265	10000000	29500000	30500000	-225117.9...	23709.92...	-242276.5...	30522.13...	-281806.2...	13093.03...	-10847.66...	11381.223...	-54559
46	-322	0.4	0.000232	0.000682	0.0283	10000000	30500000	29500000	-238405.9...	30909.05...	-253277.3...	36124.86...	-282392.5...	6841.943...	-12040.81...	7155.007...	-59584
47	-144	0.3	0.000284	0.000752	0.03	10400000	30500000	29500000	-244987.9...	0	-254951.8...	0	-285726.1...	0	-10960.73...	0	-61169
48	-500	0.5	0.000267	0.000752	0.0335	10000000	30500000	29500000	-228759.9...	42321.08...	-246816.7...	43260.01...	-280531.665	19025.3521	-11170.70...	10295.44...	-60396

(a) Bottom Side View



(b) Side View



(c) Plate Top Side View

Figure 6: Different Video Views. Our video ensemble data included three different viewpoints of the simulation. We show videos (A) generated from a bottom side view; (B) from the side; and (C) of the plate only (punch not drawn) from a top side view. In all cases, clusters are evident early in the simulation, merging into larger clusters as the simulation progresses.

used to correlate the five clusters with the initial velocity of the punch, as shown in Figure 3. (Note that in this example, the color bar used for the metadata happens to be similar to the coloring scheme in the videos, even though there is in fact no relation.)

The MDS scatter plot, video trajectories, time control, and metadata coloring provides the user with an intuitive understanding of the video ensemble as a whole. The complementary perspectives of these views along with the linked interactions combine to facilitate our design objective (T1). When a more detailed analysis is desired, the user can select individual videos for comparison using the video viewer. Videos are selected from either points in the MDS scatter plot or video trajectories. Once videos are selected they are shown in the video panel and can be synchronized with the time control or played individually, as described in task (T6). Synchronized videos are shown playing in Figure 4.

In Figure 4 we show representative videos from the three clusters (red, white, and blue). Examining frames from each video over time we can see why the clusters exist and why velocity is a driving factor in the formation of the clusters. Namely that a punch with a slow initial velocity (red cluster) fails to puncture the material, while faster initial velocities (white and blue) puncture the material at different rates. Further, any punch which fully punctures the material ends up in the same state at the end of the video, which is why the video clusters with higher punch velocities merge towards the end of the simulation.

Individual runs can be selected by clicking on points in the scatterplot, lines in the trajectory plot, or rows in the table. Groups of runs can be selected using control-click to add to an existing selection, or a rectangular rubberbanding operation can be used to select groups of colocated points. Selected runs are highlighted in the scatterplot, the trajectory plot and the table by enlarging points, intensifying lines, and darkening the row colorations, respectively. In the lower right corner of each video is the ID number of the run with an arrow pointing to a grid icon. Clicking the grid icon causes the scatterplot point to enlarge and flash, the trajectory line to be briefly drawn in isolation, and the table row to be scrolled to the top of the table and to flash.

Figure 5 is an example of using VideoSwarm to detect and investigate anomalies (T5). There are two runs colored in orange that appear at some distance from the rest of the cluster, both in the scatterplot and in the trajectory plot. This is true for all three videos. We select those two videos and another run that is part of the main cluster, but is the most distant from the two anomalous runs (so as to maximize the differences between the videos). Examining the videos, we can determine that for the two anomalous runs the stresses in the plate are much lower than for the other runs with that same initial velocity (the plate is colored by Von Mises stress, with red representing the highest stress values).

As a final remark, we note that we have three video ensembles, all generated from the same simulation data. The videos were taken from different view points, so generated different MDS scatter plots and trajectories. The three different ensembles are shown in Figure 6. In any case, the clusters in our ensemble corresponding to punch velocity were evident in all three different videos viewpoints we used.

Discussion

VideoSwarm is designed to assist scientists and engineers in understanding numerical simulations using ensembles of videos. Advantages to this approach include space savings, information beyond summary statistics, and multiple perspective analysis. However, there are disadvantages too, namely choice of variable used in the construction of the video, color mapping of the variable in the video rendering, and choice of perspective(s) used. Since we can't store all the results from every simulation, these choice have to be made even before the simulations are run, and there will always be tradeoffs involved.

Fortunately, at least one of these choices can be mitigated by the use of floating point images instead of traditional RGB images. To be more precise, we note that since we vectorize the RGB image values, it is critical that the color mapping used in the original rendering of the videos be fixed not only for the duration of the simulation, but between simulations. This is necessary to ensure that the values being compared are not shifting and that they encode the same meaning in terms of the simulation variable values. We are aware that this latter constraint will pose problems for simulations where variables have a broad dynamic range. Consequently, our future work will be to shift from RGB-valued pixels to floating point valued images.

Although VideoSwarm is designed specifically for numerical simulations, another inherent limitation in our approach is the strict adherence to time alignment. In fact, VideoSwarm is intended to be used for analyzing video data where the timing of events is as important as the event itself. Therefore, frames with identical characteristics that occur earlier or later in other simulations are regarded as being different. A different analysis technique would be required to identify shared features that are shifted in time. Additionally, because VideoSwarm does a frame-by-frame comparison, it is limited to analyzing only video ensembles containing matching numbers of frames in each video.

Finally, in our punch-plate example, the videos separated into five distinct clusters using MDS. In our other three examples (not shown), we did not observe such distinct, well-separated clusters. Instead, we saw one large group, often with distinct geometric patterns. However, by using metadata coloring, we were able to understand the major underlying features of the ensemble in each case. This is not an automatic approach, and depends on the metadata provided, so it might not always work, just as we won't always observe clusters. Thus additional future work might examine the discovery of optimal correlations between the MDS visualizations and the metadata provided.

Conclusion

Interactive visualization of ensemble data is a challenging problem. In addition to organizing huge quantities of data for display, there are many potential algorithms available for analysis. We have designed a web application to help solve these problems in the case of video ensemble data.

Our tool, *VideoSwarm*, uses a scatter plot and trajectories to abstract a video ensemble to provide an intuitive understanding of all the videos simultaneously. Although we use MDS to provide the coordinates for the scatter plot and the time series trajectory plots, other algorithms can be easily substituted. Possibilities include alternative distance metrics, alternative dimension reduction algorithms, and alternative time series type traces for a video.

Given our representation of the ensemble using the MDS scatter plot and video trajectories, VideoSwarm provides a subject matter expert with a lightweight, no-installation, browser-based interface for examining the data. VideoSwarm implements a real-time control of the MDS scatter plot as well as an interactive interface for video playback and metadata examination.

Instead of making the analyst an evaluator of the data mining results, VideoSwarm provides an easy to use interface which encourages the analyst to explore the data independently. Further, since VideoSwarm is implemented as a Slycat plugin, management of multiple datasets, multiple users, and access control is provided, encouraging collaboration between multiple analysts while maintaining data privacy.

Acknowledgments

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] Yonathan Aflalo and Ron Kimmel. Spectral multidimensional scaling. *Proceedings of the National Academy of Sciences*, 110(45):18052–18057, 2013.

[2] Oluwafemi S. Alabi, Xunlei Wu, Jonathan M. Harter, Madhura Phadke, Lifford Pinto, Hannah Petersen, Steffen Bass, Michael Keifer, Sharon Zhong, Chris Healey, and Russell M. Taylor II. Comparative visualization of ensembles using ensemble surface slicing, 2012.

[3] I. Borg and P. J. F. Goenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[4] S. Bruckner and T. Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1468–1476, Nov 2010.

[5] D. Coffey, C. L. Lin, A. G. Erdman, and D. F. Keefe. Design by dragging: An interface for creative forward and inverse design with simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2783–2791, Dec 2013.

[6] Patricia J. Crossno, Timothy M. Shead, Milosz A. Sielicki, Warren L. Hunt, Shawn Martin, and Ming-Yu Hsieh. Slycat ensemble analysis of electrical circuit simulations. In Janine Bennett, Fabien Vivodtzev, and Valerio Pascucci, editors, *Topological and Statistical Methods for Complex Data*, pages 279–294, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[7] Patricia Crossno. Challenges in visual analysis of ensembles. *IEEE Computer Graphics and Applications*, 38(2):122–131, Mar./Apr. 2018.

[8] Vin de Silva and Joshua B. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford University, 2004.

[9] Nahum Gershon. Visualization of an imperfect world. *IEEE Comput. Graph. Appl.*, 18(4):43–45, July 1998.

[10] L. Gosink, K. Bensema, T. Pulsipher, H. Obermaier, M. Henry, H. Childs, and K. I. Joy. Characterizing and visualizing predictive uncertainty in numerical ensembles through bayesian model averaging. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2703–2712, Dec 2013.

[11] Henning Griethe and Heidrun Schumann. The visualization of uncertain data: Methods and problems. In *Proceedings of SimVis '06*, pages 143–156. SCS Publishing House, 2006.

[12] M. Hlawatsch, P. Leube, W. Nowak, and D. Weiskopf. Flow radar glyphs – static visualization of unsteady flow with uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1949–1958, Dec 2011.

[13] M. Hummel, H. Obermaier, C. Garth, and K. I. Joy. Comparative visual analysis of lagrangian transport in cfd ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2743–2752, Dec 2013.

[14] C. R. Johnson and A. R. Sanderson. A next step: Visualizing errors and uncertainty. *IEEE Computer Graphics and Applications*, 23(5):6–10, Sept 2003.

[15] Christopher Johnson, Steven G. Parker, Charles Hansen, Gordon L. Kindlmann, and Yarden Livnat. Interactive simulation and visualization. *Computer*, 32(12):59–65, December 1999.

[16] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Cryst. Sect. A*, 32(5):922–923, 1976.

[17] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, Jan 2002.

[18] K. Matkovic, D. Gracanin, M. Jelovic, A. Ammer, A. Lez, and H. Hauser. Interactive visual analysis of multiple simulation runs using the simulation model view: Understanding and tuning of an electronic unit injector. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1449–1457, Nov 2010.

[19] K. Matkovic, D. Gracanin, M. Jelovic, and H. Hauser. Interactive visual steering - rapid visual prototyping of a common rail injection system. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1699–1706, Nov 2008.

[20] Jurriaan D. Mulder, Jarke J. van Wijk, and Robert van Liere. A survey of computational steering environments. *Future Gener. Comput. Syst.*, 15(1):119–129, February 1999.

[21] H. Obermaier, K. Bensema, and K. I. Joy. Visual trends analysis in time-varying ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(10):2331–2342, Oct 2016.

[22] H. Obermaier and K. I. Joy. Future challenges for ensemble visualization. *IEEE Computer Graphics and Applications*, 34(3):8–11, May 2014.

[23] M. Otto, T. Germer, and H. Theisel. Uncertain topology of 3d vector fields. In *2011 IEEE Pacific Visualization Symposium*, pages 67–74, March 2011.

[24] Mathias Otto, Tobias Germer, Hans-Christian Hege, and Holger Theisel. Uncertain 2d vector field topology. *Computer Graphics Forum*, 29(2):347–356, 2010.

[25] Alex T. Pang, Craig M. Wittenbrink, and Suresh K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, 1997.

[26] H. Piringer, S. Pajer, W. Berger, and H. Teichmann. Comparative visual analysis of 2d function ensembles. *Computer Graphics Forum*, 31(3pt3):1195–1204, 2012.

[27] Kristin Potter, Andrew Wilson, Peer-Timo Bremer, Dean Williams, Charles Doutriaux, Valerio Pascucci, and Chris Johnson. Visualization of uncertainty and ensemble data: Exploration of climate modeling and weather forecast data with integrated visus-cdat systems. *Journal of Physics: Conference Series*, 180(1):012089, 2009.

[28] Kristin Potter, Andrew Wilson, Peer Timo Bremer, Dean Williams, Charles Doutriaux, Valerio Pascucci, and Chris R. Johnson. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *2009 IEEE International Conference on Data Mining Workshops*, pages 233–240, Dec 2009.

[29] A. J. Pretorius, M. A. Bray, A. E. Carpenter, and R. A. Ruddle. Visualization of parameter space for image analysis. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2402–2411, Dec 2011.

[30] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead. Noodles: A tool for visualization of numerical weather model ensemble uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1421–1430, Nov 2010.

[31] Michael Sedlmair, Christoph Heinzl, Stefan Bruckner, Harald Piringer, and Torsten Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics*, 20:2161–2170, 2014.

[32] Swastik Singh, Song Zhang, William A. Pruett, and Robert L. Hester. Ensemble traces: Interactive visualization of ensemble multivariate time series data. In *Visualization and Data Analysis 2016, San Francisco, California, USA, February 14-18, 2016*, pages 1–9, 2016.

[33] Sierra Solid Mechanics Team. Sierra/solidmechanics 4.22 user's guide. Technical Report SAND2011-7597, Sandia National Laboratories, October 2011.

[34] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[35] J. Waser, H. Ribicic, R. Fuchs, C. Hirsch, B. Schindler, G. Bloschl, and E. Gröller. Nodes on ropes: A comprehensive data and control flow for steering ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1872–1881, Dec 2011.

[36] R. T. Whitaker, M. Mirzargar, and R. M. Kirby. Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2713–2722, Dec 2013.

[37] Craig M. Wittenbrink, Alex T. Pang, and Suresh K. Lodha. Glyphs for visualizing uncertainty in vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):266–279, September 1996.

[38] Tynia Yang, Jinze Liu, Leonard Mcmillan, and Wei Wang. A fast approximation to multidimensional scaling, by. In *Proceedings of the ECCV Workshop on Computation Intensive Methods for Computer Vision (CIMCV*, 2006.