SAND2018-1555C

VideoSwarm: Interactive Visualization of Simulation Video Ensembles

Jaxon M. Gittinger, Shawn Martin, Milosz A. Sielicki, Matthew Letter, Warren L. Hunt, Patricia J. Crossno

Sandia National Laboratories
Sandia National Laboratories, Albuquerque, NM 87185, USA

Introduction

- Ensembles are produced by varying parameters of numerical simulations for different runs.
- Used in understanding uncertainty in model parameters, behavior of the model in relation to the physical system, and the properties of the system.
- Videos take up less space and can contain more information that just summary statistics.
- VideoSwarm uses two levels of abstraction to visualize video ensembles.
- Multidimensional scaling makes two-dimensional maps of the relationship between videos.
- Maps are indexed by one-dimensional video trajectories, and computed using multidimensional scaling.

Research Questions

- How can we compare videos in our ensemble?
- Where are the videos similar and where do they diverge?
- Do the videos cluster?
- Can we identify interesting behavior in the videos and where it occurs?
- Without watching every video, how can we digest the full content of the data?

Multidimensional Scaling

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 video v_i as a vector $\boldsymbol{f}_{it} = [f_{itk}]$, 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

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

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

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

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

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

where D^2 is the componentwise square of D, and $H = I - II^T/n$, 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. Resuming our use of the parameter t, we denote by E_t the two-column matrix containing the MDS coordinates for time t.

Kabsch Algorithm

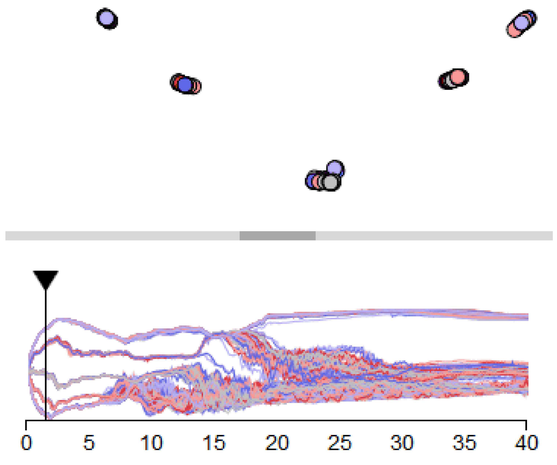
Eigenvectors computed by MDS are unique only up to sign. This fact can manifest itself as directional flips in the coordinates given even small changes in time. Kabsch algorithm computes an optimal rotation using Singular Value Decomposition to align each set of coordinates. If we assume that matrices 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 $A = U \sum V^T$. If we denote $r = sign(\det(VU^T))$ then the rotation matrix is given by

$$R = V \begin{bmatrix} 1 & 0 \\ 0 & r \end{bmatrix} U^T$$

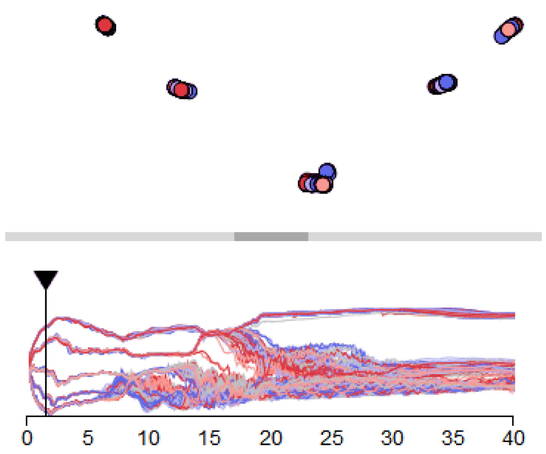
Trajectories

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.

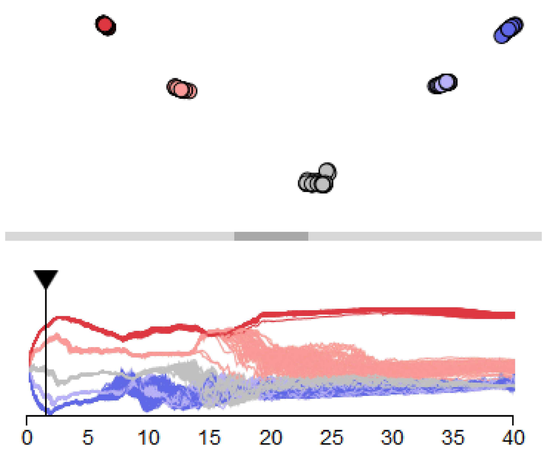
User Interface



(a) Coloring by punch/plate friction.

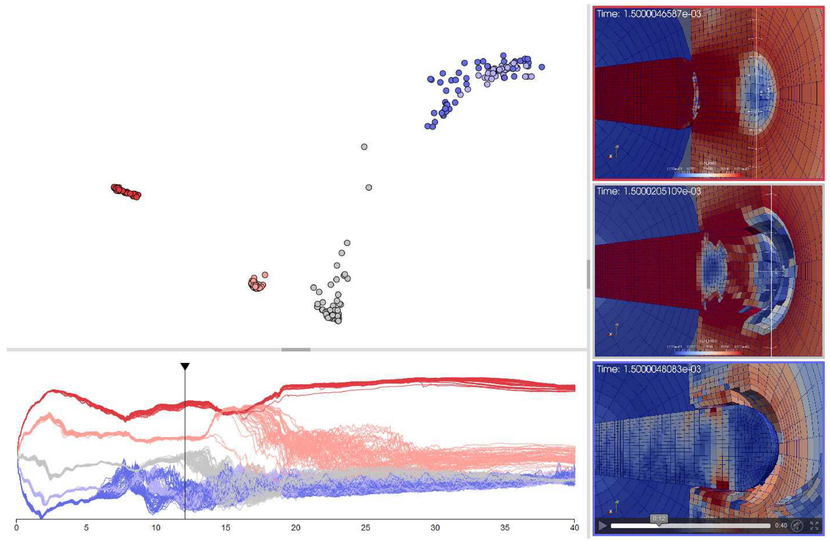


(b) Coloring by plate density.

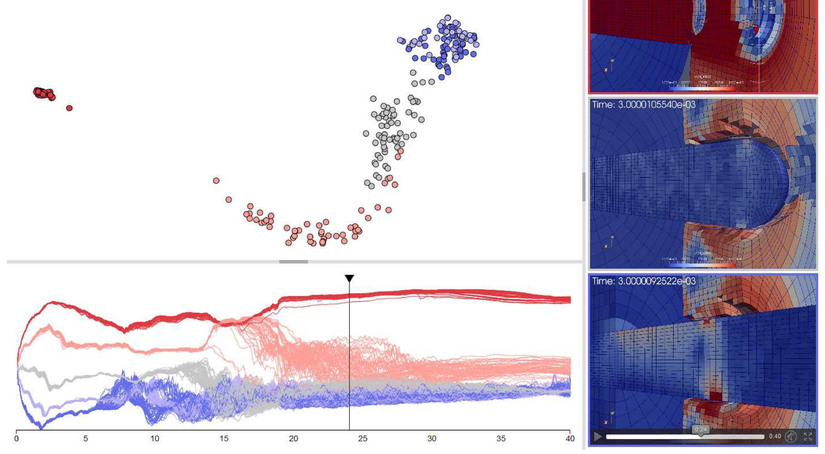


(c) Coloring by punch velocity.

Figure 1. 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. In **(a)**, we show our ensemble colored by punch/plate friction; in **(b)**, it is colored by plate density; in **(c)**, by punch velocity. From the colorings, it is clear that punch velocity is driving the formation of the clusters in the ensemble.



(a) Videos at 12 seconds



(b) Videos at 24 seconds

Figure 2. 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, gray, and blue clusters (arranged in the video pane from top to bottom). We show the videos at 12 seconds **(a)** and 24 seconds **(b)**.

Conclusion

- Interactive visualization of ensemble data is a challenging problem.
- There are huge quantities of data to display, and many potential algorithms to do analysis.
- 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 time series trajectory plots, other algorithms can be easily substituted.
- VideoSwarm provides a subject matter expert with a lightweight, no-installation, browser-based interface for examining the data.
- 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.
- Since VideoSwarm is implemented as a Slycat plugin, management of multiple datasets, multiple users, and access controls are also provided. This encourages collaboration between multiple analysts while maintaining data privacy.

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY