Ensemble Visualization For Cyber Situation Awareness of Network Security Data

Lihua Hao*
Department of Computer Science
North Carolina State University

Christopher G. Healey†
Department of Computer Science
North Carolina State University

Steve E. Hutchinson‡
ICF International
U.S. Army Research Laboratory

Abstract— Network security analysis and ensemble data visualization are two active research areas. Although they are treated as separate domains, they share many common challenges and characteristics. Both focus on scalability, time-dependent data analytics, and exploration of patterns and unusual behaviors in large datasets. These overlaps provide an opportunity to apply ensemble visualization research to improve network security analysis. To study this goal, we propose methods to interpret network security alerts and flow traffic as ensemble members. We can then apply ensemble visualization techniques in a network analysis environment to produce a network ensemble visualization system. Including ensemble representations provide new, in-depth insights into relationships between alerts and flow traffic. Analysts can cluster traffic with similar behavior and identify traffic with unusual patterns, something that is difficult to achieve with high-level overviews of large network datasets. Furthermore, our ensemble approach facilitates analysis of relationships between alerts and flow traffic, improves scalability, maintains accessibility and configurability, and is designed to fit our analysts' working environment, mental models, and problem solving strategies.

Index Terms—Ensemble, security, visualization

1 Introduction

The world is increasingly relying on computer networks. Given the proliferation of network attacks and vulnerabilities, network security analytics has become an important area of computer science, and more recently data visualization. To maintain the security and stability of a network system, analysts continuously collect vast amount of data that capture important characteristics about their networks, then analyze the data to detect attacks, intrusions, and suspicious activity hidden in the traffic. Visualization has been proposed as an important component of this effort, since it allows for interactive exploration and analysis of large amounts of data, and can help analysts detect unexpected patterns more efficiently and effectively than traditional, text-based representations [5, 25, 30].

A second area of research that has grown rapidly in recent years is ensemble analysis. In numerous disciplines, scientists collect data produced by a series of runs of a simulation or an experiment, each with slightly different initial conditions or parameterizations. This collection of related datasets—an *ensemble*—has been widely used to simulate complex systems, explore unknowns in initial conditions, investigate parameter sensitivity, mitigate uncertainty, and compare structural characteristics of models. Each individual dataset forms a *member* of the ensemble. Ensemble visualization is an active area of research in visualization, specifically designed for exploring and comparing both within and between members of massive ensemble datasets.

Even though network security data and ensemble data look quite different at first glance, they can be seen as similar in terms of their analytic challenges and goals. Both are large and time-dependent, necessitating analysis in the time dimension and approaches to support scalability. Ensemble visualization focuses on comparison and aggregation of related ensemble members, while security visualization focus on exploration of correlations between network traffic. Although visualization of scientific ensemble and network data at the most detailed level will likely be different, high-level ensemble overviews and frameworks could allow an analyst to quickly identify and drill down on subsets of interesting or suspicious network traffic. If we view network security data as a type of ensemble, there is an important opportunity to apply ongoing ensemble visualization research to improve network security analytics. This also suggests that future findings in ensemble visualization may further enhance cyber situation awareness.

To explore this hypothesis, we transformed a scientific ensemble visualization system we designed for nuclear physicists studying particle collisions to a network ensemble visualization system built to analyze and visualize alerts and flow traffic. The first requirement is to convert network data into ensemble form. In our system, a network ensemble consists of related, time-dependent sequences of alerts and flow traffic, each representing a single ensemble member. Relationships between the members are defined either by network properties or analyst-chosen time windows. The ensemble visualization next calculates dissimilarities between pairs of members using time-series comparison techniques. These dissimilarities are used to perform agglomerative clustering to combine similar members. Cluster results are visualized as a tree, representing a level-of-detail overview of inter-member relationships within the network data. The cluster tree visualization allows analysts to interactively choose subsets of members to compare, analyze and visualize, allowing them to efficiently discover traffic with similar or unique patterns. It also meets the design requirements demonstrated in our prototype security visualization system, built in the collaboration with security analysts at the U.S. Army Research Laboratory (ARL) [8]:

- visualizations must "fit" an analyst's mental models,
- visualizations must integrate into an analyst's working environment,
- visualizations must be configurable by the analyst,
- visualizations must be easily understandable to the analyst,
- visualizations must scale to large data sources, and
- visualizations must support an analyst's existing problem solving strategies.

Meeting these requirements ensures a network ensemble visualization is consistent and compatible with our existing security visualizations. More importantly, it significantly improves the likelihood that the system can provide useful information to improve cyber situation awareness, but without interrupting or conflicting with an analyst's current workflow or problem solving approaches.

[*] pku_lihuahao@hotmail.com
[†] healey@ncsu.edu
[‡] steve.hutchinson@us.army.mil

978-1-4673-7599-3/15/$31.00 ©2015 IEEE

2 Background

Visualization converts data into visual forms that support exploration, discovery, validation, presentation, and rapid and effective comprehension over large amounts of data. Network security and ensemble analysis are two research areas that benefit from this approach.

2.1 Network Security Visualization

A wide range of visualization techniques have been applied to support visual analytics of network security data. This type of data is often large, time-dependent, and contains multiple correlated data elements (e.g., alerts and flow traffic). A common technique is to start by aggregating large amounts of data into an overview, then providing additional details on demand.

Various survey papers discuss systems that use this approach (e.g., [5, 25]). Additional examples include Roberts use of bar charts and geographic heatmaps to visualize network health over time, with a reticle view to probe individual machines [22], Phan's Isis system that uses histogram timelines with event plots to visualize individual events [19], McPherson's PortVis system that summarizes data on a timeline, then allows individual ports to be explored in detail [16], Kan's NetVis system that uses treemaps to subdivide a company's network by department, then host, then alerts by host [12], Lakkaraju's NVisionIP system that visualizes an overview of network state, then collections of suspicious machines, then details on an individual machine [14], Taylor's FlowVis system that visualizes netflows as host activity plots, connected bundles of interactions between hosts, and flow data for a single host [26], Conti's use of semantic zooming, interactive encoding, and querying of network packets [4], or our own system that allows an analyst to aggregate and filter through a back-end SQL server to visualize data using different types of charts, then to drill down within the charts to reveal increasing levels of detail [8].

These types of overview+detail systems rely on analysts to identify correlations between network traffic. Detailed information is often lost in a high-level visualization, making it difficult for analysts to gain the in-depth understanding of data properties or relationships between traffic flows that are needed to help them choose more detailed views.

2.2 Ensemble Visualization

Researchers from various scientific disciplines are actively constructing *ensembles:* collections of related datasets built from a series of runs of a simulation or an experiment, each with slightly varying initial conditions or parameters. Data collected from each run—an *ensemble member*—is typically both spatial and temporal. Compared with traditional scientific data, ensembles are difficult to analyze and visualize due to their large size and high complexity [29].

Different techniques have been proposed to analyze relationships within and between members of a scientific ensemble. Many systems provide either simple overviews of an entire ensemble using aggregation, or comparative visualizations that are limited to small numbers of members. Potter's Ensemble-Vis system analyzes weather and climate models through linked views built from means and standard deviations [21]. Sanyal's Noodles system uses *spaghetti plots*, a technique for uncertainty visualization in meteorology that displays a series of contours to highlight specific attribute boundaries, together with more sophisticated statistical aggregation [23]. Alabi's ensemble surface slicing approach combines surface slices to highlight subtle variations in member surfaces [1]. Phadke's pairwise sequential animations use color, shape, and size to compare data between pairs of members [18].

More recently, Piringer designed a system to interactively compare 2D function ensembles using a domain overview and a member detail view [20]. Matkovic developed an interactive system to compare multi-run simulation results as families of 2D data surfaces [15]. Whitiker and Mirzargar proposed contour and curve boxplots to visualize statistical properties, outliers, and variability in ensembles of contours or curves [17, 28]. Band depth statistically summarizes the centrality of members of an ensemble, which are visualized using specialized boxplots. Demir developed multi-charts, an overlay of bar and line charts to present statistical properties of ensemble members [6]. Köthur studied temporal properties of ensembles, generating temporal profile clusters for different members [13], then consolidating them to identify profiles representing specific features of interest.

In this paper, we chose to apply our most recent ensemble visualization algorithms to network security data. Our system starts with an overview of inter-member relationships. We mathematically measure dissimilarity between pairs of ensemble members, applying time-series data comparison techniques when members are time dependent. We use the dissimilarity matrix to perform agglomerative clustering, then visualize the results as a level-of-detail cluster tree, where each cluster contains members that are similar. Clusters higher in the tree relax the similarity threshold required to associate their members.

Analysts interact with the cluster tree to choose which subsets of members to visualize. The key to the system is that it allows an analyst to choose how similar (or how different) a group of members must be during analysis. This removes the need for the system to choose an "appropriate" similarity threshold. It also allows the analyst to interactively vary the level-of-detail as they study their data.

2.3 Visual Perception

Stepping back from the specifics of network and ensemble visualization, design choices must be made regarding the basic presentation of different data attribute values. For example, how should properties like time, IP addresses, numbers of alerts, and time windows for netflows be represented in a visualization?

To meet the requirements of "fitting" an analyst's mental models and providing easily understandable visualizations, we chose to use basic charts as a framework for our visualizations. Line graphs, pie charts, and scatterplots are already used extensive by our analysts, and therefore are well understood.

The second goal is how to effectively integrate information into a chart. For example, consider an analyst who wants to explore the number of Snort alerts that occur between different source IP–destination IP pairs over a given time window. This can be visualized as a scatterplot with source IP on the *x*-axis, destination IP on the *y*-axis, and square "glyphs" in each source IP–destination IP cell where alerts occur. The size of the glyph represents alert frequency: larger for more alerts. This leads to two important questions: (1) Which sizes should we assign to different alert counts? and (2) Is size the best choice in this situation to visually encode the number of alerts for a given source IP–destination IP pair?

Answering these questions requires an understanding of human visual perception. Visual properties like size, color, and brightness are interpreted by the visual system first at a physical, and then at a cognitive or perceptual level [7, 27]. We have conducted numerous controlled psychophysical experiments in our laboratory to study how the visual system perceives color, texture, and motion, both in isolation and in combination with one another [9, 10, 11]. Results from these experiments provide guidelines for choosing the best combination of visual features to represent different data attributes. This allows us to design perceptually optimal visualizations, where the most perceptually salient visual features are assigned to the data properties the analyst deems most important. The choice of visual features is also guaranteed to avoid *visual interference*, a situation where less important data values can hide or obscure more important patterns and results.

Our knowledge of visual perception was applied when we chose which visual features to use to represent different data attributes, and how to map the given feature to the attribute's values (e.g., how to choose sizes that accurately represent alert courts, or colors that best differentiate line graphs representing different destination IP addresses).

3 Network Ensemble Data

To apply ensemble visualization to network security data, the first challenge is terminology mapping. How can we define a network ensemble from security datasets, and how can we divide the data into a series of related network traffic, analogous to members in an ensemble? We focus on two common types of network security data: Snort alerts and flows. The Snort alert dataset contains source and destination IP, port, time, protocol, message, and classification. The flows dataset contain source and destination IP, port, flags, start time, end time, protocol, number of packets, and number of bytes. An alert belongs to a flow if it is detected within the time range of the flow and has the same source and destination IP. Given these input sources, we support analysis and visualization of two types of ensembles: an alert ensemble and and a flow ensemble.

To maintain the design requirement of configurability, we propose a general framework to construct a network data ensemble. This allows analysts to configure details of the ensemble definition, if they choose to do so. A network ensemble consists of related network traffic (analogous to ensemble members), each representing a temporal sequence of alerts or flows that fall within equal-sized time windows. Analyzing relationships (similarities) between this network traffic is one important goal of cyber situation awareness.

Specifically, we offer two ways to define members in a network ensemble. The first method correlates data properties to identify members. For example, we could define source IP and source port to specify members. Now, alerts or flows sent from each source IP/port combination form a network ensemble member. The second method divides the ensemble time window into a number of smaller time windows, each identifying a member in the ensemble. For example, if the dataset consists of network traffic for a 24-hour period, hourly traffic can represent an ensemble member. Finally, analysts can control the data values stored within each member. For example, we can analyze inter-member relationships in an alert ensemble by comparing changes in the of numbers of alerts over the time dimension.

4 Visualization Design

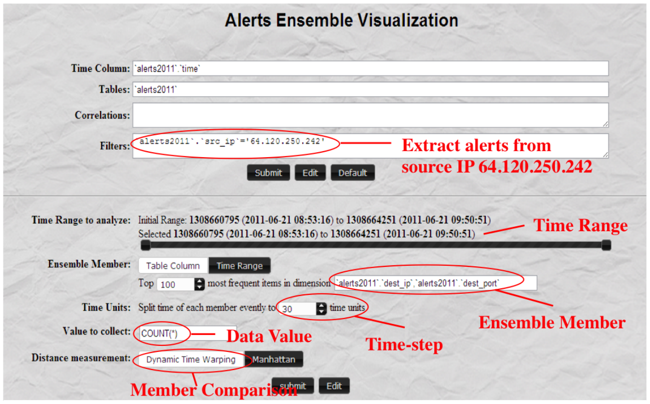
Our ensemble visualization system is built to meet the design requirements outlined in the Introduction. Based on this, we implemented the system as a web application, to integrate it into our analysts' working environment, using HTML5 and Javascript for the user interface and visualization components, and MySQL and PHP for server-side data management. This approach is compatible with our security visualization system discussed in [8]. Our approach supports configurability by allowing analysts to build an alert or flow ensemble that meets their specific task requirements. Visualizations are based on 2D charts, providing accessibility through a common visualization framework. Level-of-detail techniques allow us to scale to large network datasets. Results suggest it is feasible to employ ensemble visualization for analyzing network traffic, and to integrate this approach into existing network security visualization systems in ways that maintain flexibility and effectiveness.

4.1 Analyst-Driven Ensemble Definition

In scientific domains, an ensemble is collected as a set of related datasets, each forming an ensemble member of data collected at different time-steps. Investigation of individual members and comparison between members are two of the most important analysis tasks. To perform ensemble visualization on network security data, the first and most essential step is to structure the network data as an ensemble, based on the needs of network traffic analysis and comparison.

There are different ways to view network data as an ensemble. We support analysis of two types of network data—alert ensembles and flow ensembles. The user interfaces to configure alert and flow ensembles are nearly identical, with only slight variations in alert or flow-specific details.

Fig. 1 shows the interface used to define an alert ensemble. Network data is contained in multiple SQL tables, forming a large number of data columns. Since the ensemble may only use a few columns and a subset of the data, an analyst can define the SQL tables that contain the alert data of interest, the time dimension column, correlations between the tables (if more than one table is selected), and any additional constraints to form the ensemble. The system will automatically identify the time window that covers the ensemble's data. The analyst can choose one or more table columns to define ensemble members, or evenly subdivide the ensemble's time window into a number of smaller time periods, each representing a member.



Alerts Ensemble Visualization

Time Column: `alerts2011`.`time`

Tables: `alerts2011`

Correlations:

Filters: alerts2011`.`src_ip`='64.120.250.242'

Extract alerts from source IP 64.120.250.242

Time Range to analyze: Initial Range: 1308660795 (2011-06-21 08:53:16) to 1308664251 (2011-06-21 09:50:51)
Selected 1308660795 (2011-06-21 08:53:16) to 1308664251 (2011-06-21 09:50:51)

Time Range

Ensemble Member: Table Column | Time Range
Top 100 most frequent items in dimension `alerts2011`.`dest_ip`,`alerts2011`.`dest_port`

Ensemble Member

Time Units: Split time of each member evenly to 30 time units

Time-step

Value to collect: COUNT(*)

Data Value

Distance measurement: Dynamic Time Warping | Manhattan

Member Comparison

Fig. 1: A user interface to to define an alerts ensemble

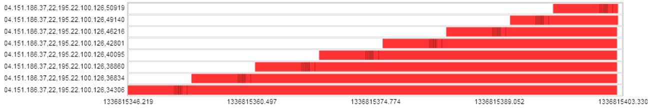


Fig. 2: Flow ensemble, time on the *x*-axis, destination IP plus port on the *y*-axis, each red bar represents an individual flow's start and end time, black tick marks within a flow identify Snort alerts

To analyze relationships between members in an alert ensemble, we subdivide the time range of every member into a user-specified number of time-steps and aggregate alerts within each time-step. In Fig. 1 the alert ensemble contains alert data sent from source IP `64.120.250.242`. Each destination IP+port combination forms a member in the ensemble. The time window is divided into 30 time-steps, with the number of alerts calculated at every time-step. Comparing alert members is performed by comparing changes in the number of alerts over time.

The user interface to define a flow ensemble is very similar. Flow ensembles contain both flow and alert data (Fig. 2). An analyst chooses the SQL tables that contain the data, defines how to correlate alerts and flow traffic, identifies time dimension columns for the alerts and flows, and provides any additional constraints to extract the data to analyze. A flow has start and end times and contains a number of alerts, so comparing pairs of flows is more complicated than comparing numbers of alerts. Instead of aggregating data across time-steps, we view every member in a flow ensemble as a sequence of individual flows. Calculating member similarity correlates flows between pairs of members prior to comparison using dynamic time warping.

4.2 Member Comparison

One advantage of ensemble visualization is its ability to focus on relationships between ensemble members. This is often important during exploration to identify related or unusual members (e.g., network traffic). To provide an overview of inter-member relationships, we must measure the dissimilarity between members. We chose Manhattan distance to compare members that are exactly aligned in the time dimension, and dynamic time warping [2, 24] to build an optimal matching between members that may not be aligned over time.

4.2.1 Alert Member Comparison

A member m_i in an alert ensemble is a sequence of aggregated values (i.e., a number of alerts) collected at t analyst-specified time-steps $m_i = (n_{i,1}, n_{i,2}, \dots, n_{i,t})$.

A simple comparison of alert members uses Manhattan distance, where we assume exact alignment in the time dimension. Let $dis_{i,j}$ be the dissimilarity between members m_i and m_j. The Manhattan distance is calculated in $\mathscr{O}(t)$ time as:

$$dis_{i,j} = \sum_{p=1}^{t} \left| n_{i,p} - n_{j,p} \right| / t \tag{1}$$

Unfortunately, real-world network traffic is not necessarily an ideally aligned ensemble. Changes in alert traffic (e.g., the numbers of alerts) can happen at different time-steps, and across different periods of time. To shift or warp over time to get the "best" alignment, we use dynamic time warping (DTW) [2, 24]. DTW is designed to identify an optimal matching between two temporal sequences that vary in time and velocity. It compares sequences by finding a non-linear alignment that minimizes the summed distance between members in the sequences. This is done by calculating the minimum distance between m_i and m_j to form a $t \times t$ matrix W where $W(u,v)$ encodes the dissimilarity between $n_{i,u}$ and $n_{j,v}$ (i.e., the u-th time-step of m_i and the v-th time-step of m_j). Dynamic programming is then used to build the shortest warping path through W.

It requires $\mathscr{O}(t^2)$ time and space to calculate the DTW distance between m_i and m_j. Let $D'(m_{i,u}, m_{j,v})$, $1 \leq u, v \leq t$, be the minimum distance between two subsequences $m_{i,u} = (n_{i,1}, n_{i,2}, \ldots, n_{i,u})$ and $m_{j,v} = (n_{j,1}, n_{j,2}, \ldots, n_{j,v})$. The DTW dissimilarity $dis_{i,j} = D'(m_{i,t}, m_{j,t})/t$ between m_i and m_j is calculated via dynamic programming on the recursion:

$$\begin{aligned} D'(m_{i,u}, m_{j,v}) = dis(n_{i,u}, n_{j,v}) + \min\big(D'(m_{i,u-1},\\ m_{j,v}), D'(m_{i,u-1}, m_{j,v-1}), D'(m_{i,u}, m_{j,v-1})\big) \end{aligned} \tag{2}$$

where

$$dis(n_{i,u}, n_{j,v}) = \left| n_{i,u} - n_{j,v} \right| \tag{3}$$

4.2.2 Flow Member Comparison

A member m_i in a flow ensemble is a sequence of l_i flows $m_i = (f_{i,1}, f_{i,2}, \ldots, f_{i,l_i})$. Each flow has start time t_s^i and end time t_e^i, and contains zero or more alerts. This makes the comparison of flow traffic more complicated than aggregated alerts. Manhattan distance is not applicable for flow sequence comparison because the sequences may have different lengths and may not be aligned in time. To calculate the DTW distance between flow members, we must first calculate the dissimilarity between pairs of flows. Let $dis(f_u, f_v)$ be the dissimilarity between flows f_u and f_v. We propose a simple flow comparison that calculates $dis(f_u, f_v)$ based on three metrics:

1. **Duration.** Given f_i's duration $dur_i = t_e^i - t_s^i$, the duration dissimilarity between f_u and f_v is

$$dis_{\text{dur}}^{u,v} = \frac{|\, dur_u - dur_v \,|}{\max(dur_u, dur_v)} \tag{4}$$

2. **Density.** Given f_i containing n_i alerts, the density of alerts in f_i is $den_i = n_i / dur_i$. The density dissimilarity between f_u and f_v is

$$dis_{\text{den}}^{u,v} = \frac{|den_u - den_v|}{\max(den_u, den_v)} \tag{5}$$

3. **Distribution.** Given start and end times t_s^i and t_e^i for a flow f_i that receives n_i alerts at times $t_1^i, t_2^i, \ldots, t_{n_i}^i$, the intervals between alerts are $I_i = \{t_s^i - t_1^i, t_2^i - t_1^i, \ldots, t_e^i - t_{n_i}^i\}$. We use $\sigma_i = \sum_{i=1}^{n_i} (I_i - I_\mu)^2 / n_i$, the variance of the intervals between alerts, to compute distribution dissimilarity between f_u and f_v as

$$dis_{\text{dist}}^{u,v} = \frac{|\sigma_u - \sigma_u|}{\max(\sigma_u, \sigma_v)} \tag{6}$$

The individual dissimilarities are averaged and normalized to generate an overall dissimilarity between f_u and f_v. We allow an analyst to tune the weights w_{dur}, w_{den}, and w_{dist} during averaging:

$$\begin{gathered} dis(f_u, f_v) = w_{\text{dur}} dis_{dur}^{u,v} + w_{\text{dens}} dis_{\text{dens}}^{u,v} + w_{\text{dist}} dis_{\text{dist}}^{u,v} \\ 0 \leq w_{\text{dur}}, w_{\text{dens}}, w_{\text{dist}} \leq 1 \\ w_{\text{dur}} + w_{\text{dens}} + w_{\text{dist}} = 1 \end{gathered} \tag{7}$$

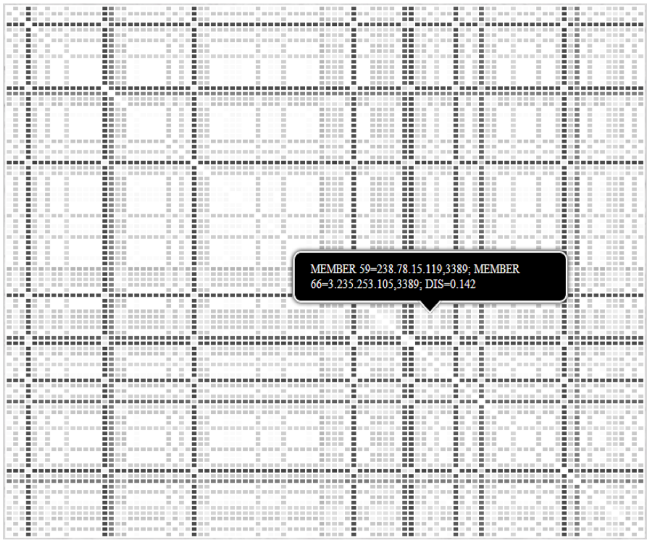


Fig. 3: Dissimilarity matrix for a 100-member alert ensemble, darker for more dissimilar

The DTW recursion is equivalent to Eq. 3, substituting $dis(f_u, f_v)$ for $dis(n_{i,u}, n_{j,v})$.

4.3 Inter-Member Relationships

For an alert or flow ensemble with N members, member comparison (Section 4.2) produces an $N \times N$ dissimilarity matrix encoding differences between all member pairs. Fig. 3 is a grayscale visualization of an alert ensemble dissimilarity matrix with $N = 100$ members. Luminance is used here, since it is one of the most perceptually salient visual properties, and because it allows viewers to identify subtle differences in the underlying data values. The brightness of cell (i, j) represents the DTW dissimilarity between the i-th and j-th members. Darker cells indicate larger dissimilarities. The matrix in Fig. 3 has a large number of white and light gray cells. This highlights that a large number of the ensemble members are very similar to one another. The small number of black or dark gray columns (or rows) show that some alert members are very different from all the others, however.

The dissimilarity matrix visualization presents a high-level overview of the relationships between members, but it is not detailed enough for analysts to quickly choose useful subsets of members. To support more in-depth exploration of the relationships between members, we use the dissimilarity matrix to perform agglomerative clustering, organizing similar alerts or flow traffic into groups according to their optimal DTW matchings. Agglomerative clustering works in a bottom-up fashion, first assigning each member to its own cluster, then iteratively merging the two most similar clusters and updating the dissimilarity matrix. This produces a level-of-detail clustering result that assign members into $k = N, k = N-1, \ldots, k = 1$ clusters. Fig. 4 shows a cluster tree visualization of agglomerative clustering on the dissimilarity matrix in Fig. 3. Orange nodes represent the new cluster created at each level. Alerts members that are very different from the others remain separated until close to the root of the tree (e.g., the members circled by the red oval).

One important issue is the ability to scale the cluster tree to large numbers of alerts or flows. There are different ways to do this. One easy alternative is to combine more members at each agglomerative step: for example, the four most similar members rather than the two most similar. Another approach, which we have found effective, is to allow analysts to choose subsets of the cluster tree by selecting start and end levels with a slider above the tree (Fig. 4). This allows them to vary the level of detail within each cluster in the visualization. A selection near the top of the tree produces a small number of clusters, where each cluster contains numerous members that span a wide similarity range. A selection near the bottom of the tree will produce

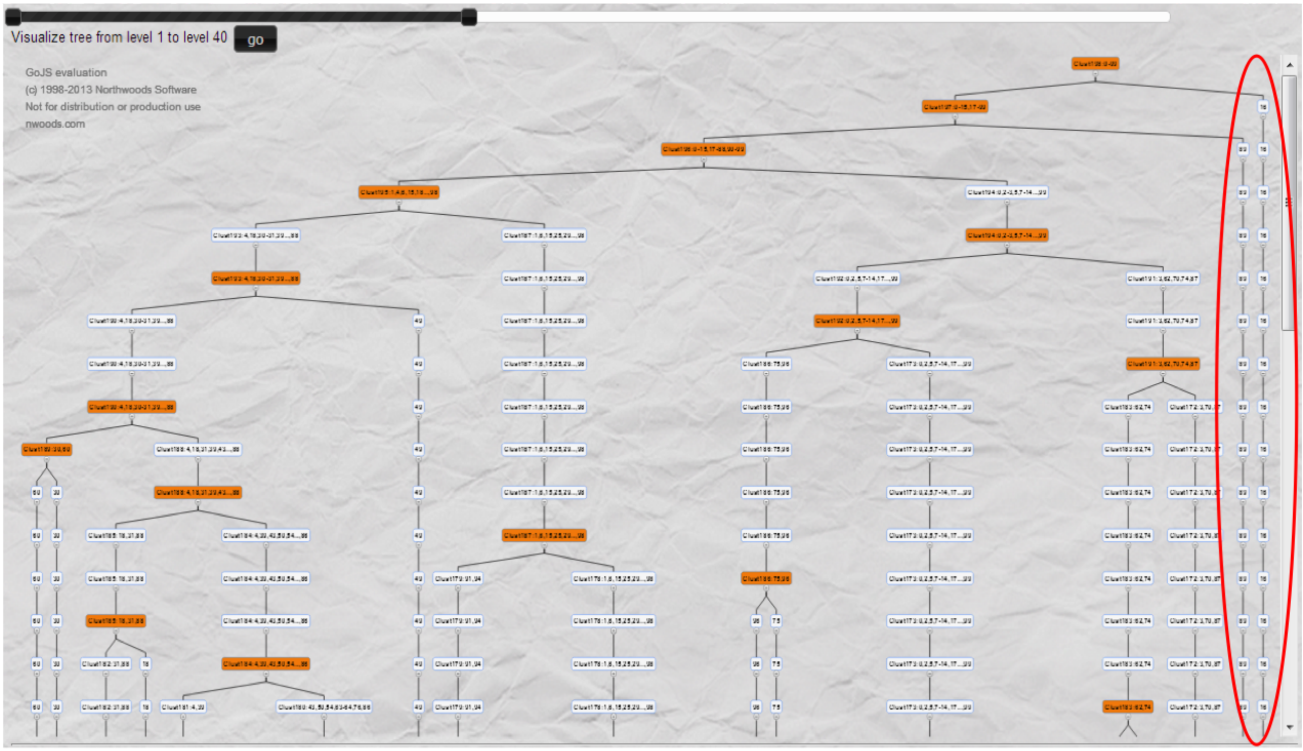

Visualize tree from level 1 to level 40
go
GoJS evaluation
(c) 1998-2013 Northwoods Software
Not for distribution or production use
nwoods.com

Fig. 4: Cluster tree for the alert ensemble in Fig. 3

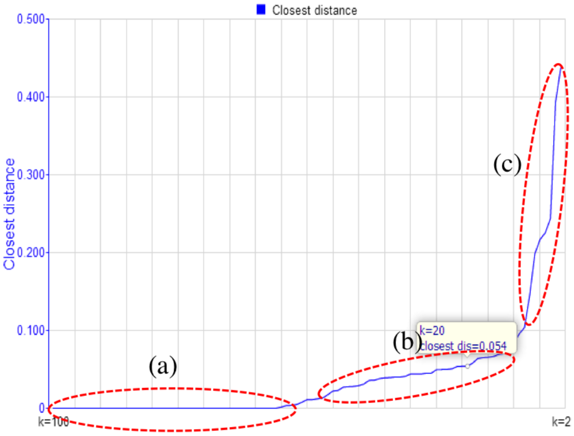


Fig. 5: Smallest dissimilarity at each level of the cluster tree

smaller clusters containing members that are very similar, but at the expensive of a large tree with more total clusters. The ability to control the initial clustering, and the subset of the cluster tree to explore, fits the requirement of allowing analysts to dynamically configure their visualizations.

To make it easier for analysts to choose where in the cluster tree they want to explore, we visualize the smallest dissimilarity (i.e., the smallest $dis_{i,j}$ or $dis(f_i, f_j)\,\forall i, j$) at each k. Fig. 5 visualizes the smallest dissimilarity at each level of the cluster tree in Fig. 4, showing three general phases:

1. As clustering begins, many members contain the same number of alerts in their time windows, producing a dissimilarity of zero for the two most similar clusters.
2. In the middle levels, clustering has combined all equivalent members, so new members are generated with different numbers of alerts from one another.
3. As clustering ends at the top few levels in the cluster tree, members with very different numbers of alerts are clustered.

Analysis of inter-member relationships in alert or flow ensembles is the same. Prior to detailed visualizations of the alerts or flow traffic, analysts use the level-of-detail cluster tree to select clusters that combine similar alerts or flow members in ways that match their needs. The cluster tree overview allows analysts to quickly choose different subsets of alerts or network traffic to compare, analyze and visualize based on similarity.

4.4 Ensemble Member Visualization

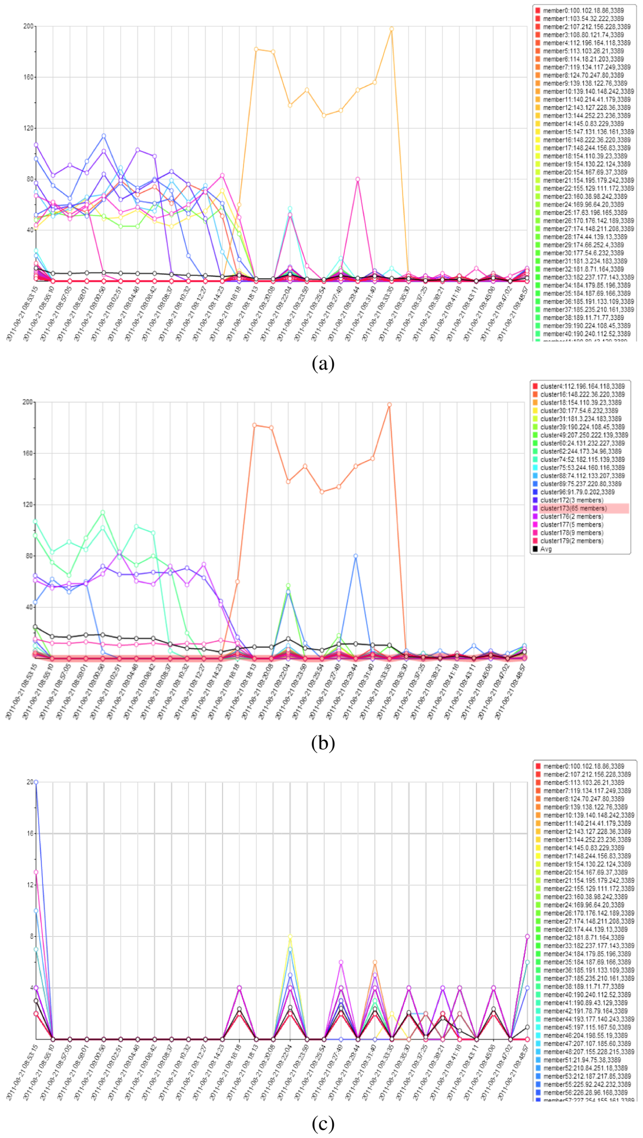
Ensemble visualization often contains detailed representations of one or more ensemble members. These detailed member visualizations may be very different depending on the types of members contained in a ensemble. To maintain consistency with our existing network security visualization system [8], we use 2D charts to visualize network traffic. Specifically, we generate line charts to visualize members in an alert ensemble and Gantt charts for members in a flow ensemble.

4.4.1 Alert Member Visualization

An alert ensemble member is a sequence of aggregated alert counts calculated at every time-step. Fig. 6a visualizes the 100-member alert ensemble shown in Fig. 3. Time unfolds on the x-axis, and the number of alerts at each time step is plotted on the y-axis. Color represents the destination IP of each alert's parent flow.

Color was chosen since it is perceptually effective at identifying nominal values that are categorized to differentiate them from one another. Here, each destination IP is viewed as a unique category. One issue to consider is that color can be perceived as ordered by a viewer. To address this, we assigned clusters in order to the common rainbow color map that runs from red, through orange, yellow, and green, ending with blue and purple. Such a color map is not appropriate for visualizing continuous values, since it includes perceived color imbalances and false color boundaries [3]. As noted above, however, it is well suited to categorical data.

Fig. 6a highlights a number of similar patterns: flows that start with numerous alerts but end with few alerts; flows with two spikes in alerts



(a)

(b)

(c)

Fig. 6: Alert member visualization: (a) a 100-member ensemble; (b) visualization of members assigned to $k = 20$ clusters; (c) visualizing each alert member in a cluster containing 65 members with similar alert patterns, but from different flows

near their center; and an outlier flow (in orange) with numerous alerts over its center.

Given the cluster tree in Fig. 4 and the dissimilarity visualization in Fig. 5, an analyst could choose to assign the 100 alerts members to $k = 20$ clusters. Fig. 6b visualizes the 20 clusters, averaging the number of alerts within each cluster at every time-step. It provides a general understanding of changes in the numbers of alerts over time. The highlighted purple line at the bottom of the graph represents a large cluster (cluster 173) that contains 65 members. A follow-on visualization of this cluster's members (Fig. 6c) confirms that changes in the number of alerts over time among the 65 members are similar.

4.4.2 Flow Member Visualization

Similar to [8], we use Gantt charts to visualize flow traffic and associated alerts. The x-axis represents time, and the y-axis represents member (e.g., a combined IP plus port). Flows are visualized as colored bars with endpoints at the flow's start and end times. Alerts appear as black vertical tick marks at the time the alert was detected.

Analysts choose clusters of flow members to visualize using the cluster tree visualization. Fig. 7 visualizes two clusters, each with two flow members. Color (red and blue) identify the two clusters, and each row represents a flow member. Zooming into a group of correlated flows from different members produces a detailed visualization that shows the flows in each cluster are similar based on time duration, alert density, and alert distribution. Without DTW, the flows in the red cluster would not be considered similar, since they start and end at different times.

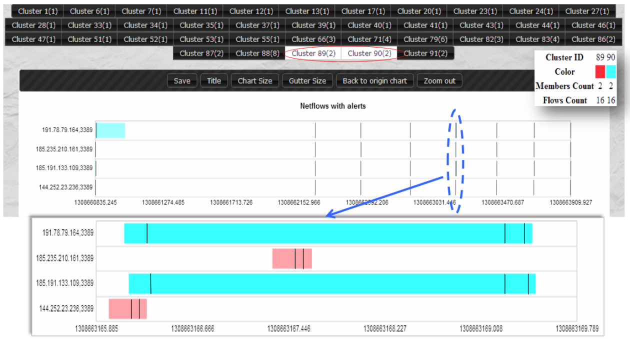


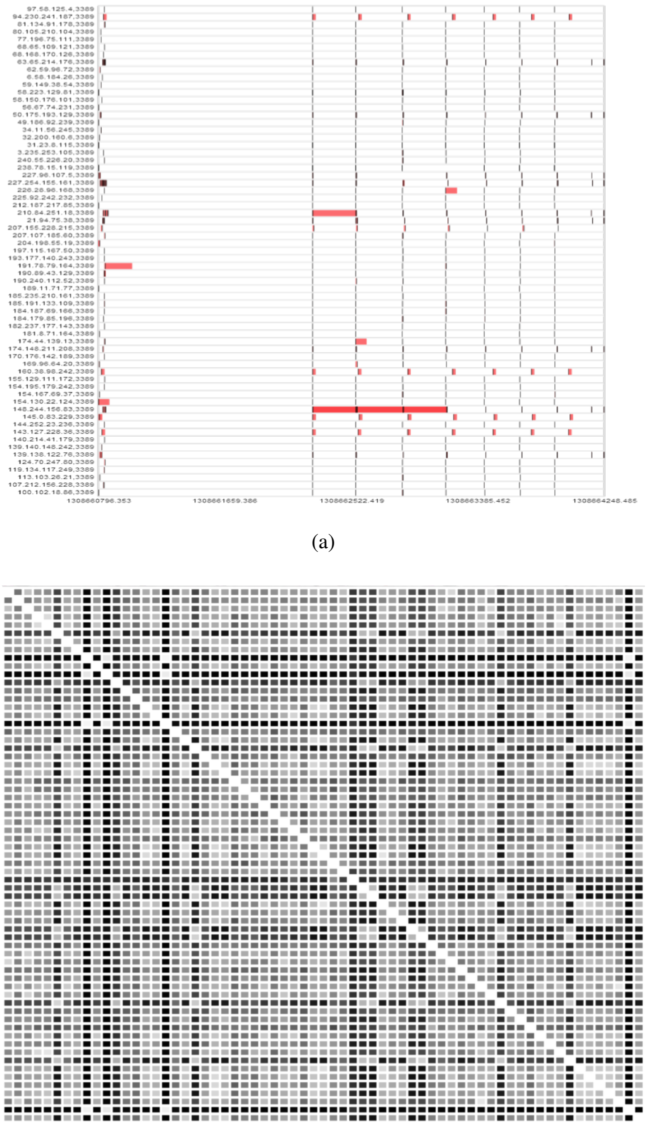
Fig. 7: Visualization of four flow ensemble members from two different clusters shown in blue and red

5 Example Analysis Session

We built a web-based system that implements our ensemble visualization framework for network security analytics. Consistent with our previous network data visualization system [8], data management occurs on a remote server running MySQL and PHP. The visualization is based on interactive 2D charts using HTML5 and Javascript. We used anonymized Snort alert and netflow traffic from one floor of our Computer Science building to test our system on real-world alert and flow patterns.

Section 4 illustrates an ensemble visualization of alerts data. Unfortunately, it is currently not possible for us to interact directly with the network security analysts at ARL. To address this, we took our alerts dataset and recruited a network security researcher in our department to act in the role of an analyst. His goal was to identify destination IPs that contain similar patterns of alerts for a target source IP. The analyst starts by selecting the SQL database and tables containing the alerts data, and setting constraints (source IP `64.120.250.242`) to filter the alerts and build an alert ensemble (Fig. 1). The analyst chooses destination IP and port to define alerts sent to a common destination as a member in the ensemble. Since the analyst is not interested in traffic with a small number of alerts, they sort ensemble members by number of alerts and analyze only the top 100 members. The system generates SQL queries to extract the relevant data and calculate dissimilarities between pairs of members. It generates a dissimilarity matrix visualization (Fig. 3), an agglomerative cluster tree visualization (Fig. 4) and a line chart characterizing dissimilarity for different numbers of clusters k (Fig. 5). Based on the overview of inter-member relationships, the analyst combines the 100 members into $k = 20$ clusters (Fig. 6b). Fig. 6c visualizes the largest cluster, containing 65 members with similar changes in the number of alerts over time. Ensemble visualization makes it easier to automatically detect similar changes in alerts patterns, something that is not as obvious from a general visualization of alert traffic.

To gain a more in-depth understanding of the alert traffic covered by the 65-member cluster, the analyst takes flow traffic into consideration by requesting a flow ensemble visualization. The alert visualization system exports SQL queries to extract the alerts in the 65-alert member. The analyst uses these constraints to retrieve associated flows to define a flow ensemble. Members in the ensemble are also defined by destination IP and port. Equal weights are assigned to the three metrics w_{dur}, w_{dens}, and w_{dist} during flow comparison (Eq. (7)). In this way, every member in the flow ensemble is correlated with the member in



(a)

(b)

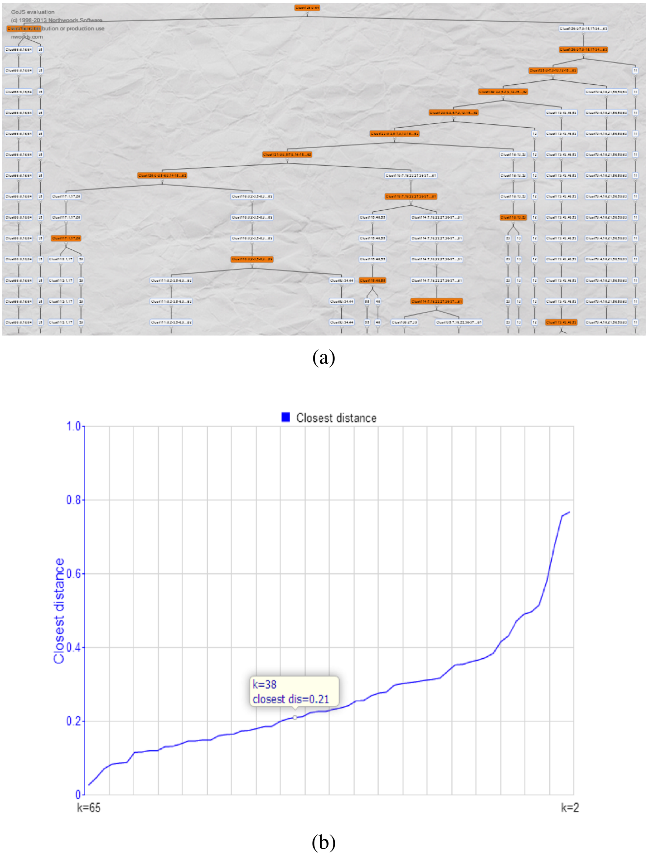
Fig. 8: Flow ensemble: (a) visualizing ensemble members in a Gantt chart; (b) dissimilarity matrix for the 65 members

the alert ensemble that is sent to the same destination.

Fig. 8a visualizes the 65 members in the flow ensemble as 65 individual Gantt charts with time on the x-axis and destination IP and port on the y-axis. At this level of detail, flow traffic for most members looks similar, making it difficult to visually distinguish the flows without zooming in. The dissimilarity matrix (Fig. 8b) indicates that the network traffic sent to different destinations are further differentiated when we include the flow data (i.e., there are more dark cells versus the dissimilarity matrix in Fig. 3).

Fig. 9 visualizes the cluster tree and the dissimilarity graph for each cluster tree level. The analyst decides that members of the ensemble are similar if their dissimilarities are smaller than 0.21. This combines the 65 members into $k = 38$ clusters. As expected, flows in members from the same cluster are similar. For example, in Fig. 10, the flows in a cluster with six members have very similar density and distributions of alerts, and relatively similar durations (as shown in the top-right overview visualization).

The analyst concludes that this particular pattern of alerts is a candidate for further, more detailed investigation. At this point he would turn to additional analysis tools or scripts that are specifically designed to investigate these types of alert patterns. The strength of the visualization is its ability to focus the analyst on a small, manageable subset of alerts and flows for follow-on analysis.



(a)

(b)

Fig. 9: Flow ensemble cluster tree: (a) cluster tree visualization; (b) dissimilarity graph at each level of the cluster tree

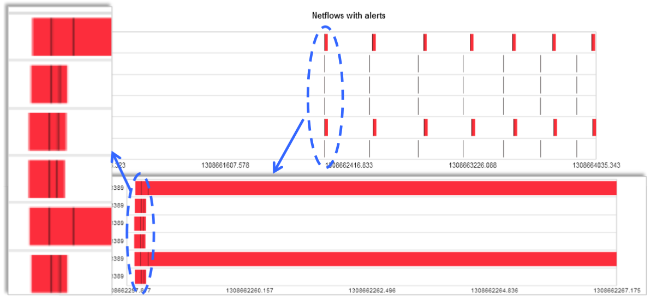


Fig. 10: A cluster of six members in the flows ensemble. Corresponding flows has very similar patterns

6 Conclusion and Future Work

We propose an ensemble visualization system to enhance cyber situation awareness of network security data. This is based on the hypothesis that network security data and ensemble data have many similar characteristics and face similar challenges. We see an important opportunity to build a connection between these two areas, allowing us to leverage on-going and future ensemble visualization techniques to enhance network security analytics. To test this hypothesis, we applied an ensemble visualization framework to the network security domain. Our network ensemble visualization system is implemented as a web-based application that is compatible with our existing security visualization system, and consistent with the requirements of our ARL colleagues.

We define a terminology mapping to transform network security data into an ensemble format. An ensemble of network data contains a number of alerts or flow traffic, represented as members in the ensemble. We provide analysts with the flexibility to configure how alerts or flow traffic are converted into members within the ensemble. Dynamic time warping is used to align ensemble members. Different metrics are then employed to compare pairs of alerts or flows. Based on the dissimilarities that result, clustering is applied to produce a level-of-detail cluster tree overview that highlights inter-member relationships in both alert and flow ensembles. Analysts interact with the cluster tree to more efficiently choose subsets of related alerts or flow traffic for detailed visualization and analysis. To provide accessibility, members in a network data ensemble are visualized as 2D charts. Ensemble visualization improves scalability in security analytics, allowing analysts to start from a large amount of alert or flow data, and rapidly drill down to a useful subset of interest. The system is designed to provide additional useful information without interfering with an analyst's current problem solving strategies and work flows.

Future work focuses on enhancing our collaborations with network analysts to tune details of the ensemble visualization. We intend to explore more flexible analyst-controlled definitions of network ensembles that are not limited to alerts and flows traffic. The current comparisons are fairly simple, so we will coordinate with network analysts to explore more sophisticated comparison metrics that improve the quality and practicality of inter-member relationship analysis. We also intend to design visualizations that summarize key features and highlight differences in a cluster of network traffic. Currently, the visualization of alert and flows ensembles are managed separately from our flexible network data visualizations, even though they are consistent with one another. We will improve the user interface to integrate the ensemble visualization system with the general security visualization system. We also intend to employ additional ensemble visualization techniques, to further demonstrate the feasibility of applying ensemble visualization to enhance cyber situation awareness.

7 Acknowledgments

This work was supported by an ARO MURI on Computer-Aided Human Centric Cyber Situation Awareness.

References

[1] O. Alabi, X. Wu, J. Harter, M. Phadke, L. Pinto, H. Petersen, S. Bass, M. Keifer, S. Zhong, C. G. Healey, and R. M. Taylor II. Comparative visualization of ensembles using ensemble surface slicing. In *Visualization and Data Analytics*, volume 8294, pages 0U: 1–12, San Francisco, CA, 2012.

[2] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD-94: AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 359–370, Seattle, WA, 1994.

[3] D. Borland and R. M. Taylor. Rainbow color map (still) considered harmful. *IEEE Computer Graphics & Applications*, 27(2):14–17, March 2007.

[4] G. Conti, J. Grizzard, M. Ahamad, and H. Owen. Visual exploration of malicious network objects using semantic zoom, interactive encoding and dynamic queries. In *Proceedings of the 2nd International Workshop on Visualization for Cyber Security (VizSec 2005)*, pages 83–90, Minneapolis, MN, 2005.

[5] T. K. Dang and T. T. Dang. A survey on security visualization techniques for web information systems. *International Journal of Web Information Systems*, 9(1):6–31, 2013.

[6] I. Demir, C. Dick, and R. Westermann. Multi-charts for comparative 3D ensemble visualization. *IEEE Transactions on Visualization and Computer Graphics*, 20:2713–2722, 2014.

[7] A. S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1995.

[8] L. Hao, C. G. Healey, and S. E. Hutchinson. Flexible web visualization for alert-based network security analytics. In *Proceedings of the 10th Workshop on Visualization for Cyber Security (VizSec 2013)*, pages 33–40, Atlanta, GA, 2013.

[9] C. G. Healey and J. T. Enns. Large datasets at a glance: Combining textures and colors in scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):145–167, 1999.

[10] C. G. Healey and J. T. Enns. Attention and visual memory in visualization and computer graphics. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1170–1188, 2012.

[11] D. E. Huber and C. G. Healey. Visualizing data with motion. In *Proceedings of the 16th IEEE Visualization Conference (Vis 2005)*, pages 527–534, Minneapolis, Minnesota, 2005.

[12] Z. Kan, C. Hu, Z. Wang, G. Wang, and X. Huang. NetVis: A network security management visualization tool based on treemap. In *Proceedings of the 2nd International Conference on Advanced Computer Control (ICACC 2010)*, pages 18–21, Shenyang, China, 2010.

[13] P. Köthur, M. Sips, H. Dobslaw, and D. Dransch. Visual analytics for comparison of ocean model output with reference data: Detecting and analyzing geophysical processes using clustering ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19:1893–1902, 2013.

[14] K. Lakkaraju, W. Yurcik, and A. J. Lee. NVisionIP: NetFlow visualizations of system state for security situational awareness. In *Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC '04)*, pages 65–72, Washington, DC, 2004.

[15] K. Matkovic, D. Gracanin, B. Klarin, and H. Hauser. Interactive visual analysis of complex scientific data as families of data surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1351–1358, 2009.

[16] J. McPherson, K. Ma, P. Krystosk, T. Bartoletti, and M. Christensen. PortVis: A tool for port-based detection of security events. In *Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC '04)*, pages 73–81, Washington, DC, 2004.

[17] M. Mirzargar, R. T. Whitaker, and R. M. Kirby. Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE Transactions on Visualization and Computer Graphics*, 20:2654–2663, 2014.

[18] M. Phadke, L. Pinto, O. Alabi, J. Harter, R. M. Taylor II, X. Wu, H. Petersen, S. A. Bass, and C. G. Healey. Exploring ensemble visualization. In *Visualization and Data Analysis*, volume 8294, pages 0B: 1–12, San Francisco, CA, 2012.

[19] D. Phan, J. Gerth, M. Lee, A. Paepcke, and T. Winograd. Visual analysis of network flow data with timelines and event plots. In *Proceedings of the 4th International Workshop on Visualization for Cyber Security (VizSec 2007)*, pages 85–99, Sacramento, CA, 2007.

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

[21] K. Potter, A. Wilson, V. Pascucci, D. Williams, C. Doutriaux, P.-T. Bremer, and C. Johnson. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *IEEE International Conference on Data Mining Workshops (ICDMW '09)*, pages 233–240, Miami, FL, 2009.

[22] J. C. Roberts, W. J. Faithfull, and F. C. B. Williams. SitaVis—Interactive situation awareness visualization of large datasets. In *Proceedings of the 2012 Conference on Visual Analytics Science and Technology (VAST 2012)*, pages 273–274, Seattle, WA, 2012.

[23] 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, 2010.

[24] P. Senin. Dynamic time warping algorithm review. Technical report, Department of Computer Science, University of Hawai'i at Mānoa, Honolulu, HI, 2008.

[25] H. Shiravi, A. Shiravi, and A. Ghorbani. A survey of visualization systems for network security. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1313–1329, 2012.

[26] T. Taylor, D. Paterson, J. Glanfield, G. C., S. Brooks, and J. McHugh. FlowVis: Flow visualization system. In *Proceedings of the Cybersecurity Applications & Technology Conference For Homeland Security 2009 (CATCH '09)*, pages 186–198, Washington, DC, 2009.

[27] C. Ware. *Information Visualization: Perception for Design, 3rd Edition*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 2012.

[28] 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*, 19:2713–2722, 2013.

[29] A. Wilson and K. Potter. Toward visual analysis of ensemble data sets. In *Proceedings of the 2009 Workshop on Ultrascale Visualization (UltraVis '09)*, pages 48–53, Portland, OR, 2009.

[30] Y. Zhang, Y. Xiao, M. Chen, J. Zhang, and H. Deng. A survey of security visualization for computer network logs. *Security and Communication Networks*, 5(4):404–421, 2012.