

# Real-Time Visualization of Streaming Text with a Force-Based Dynamic System

Jamal Alsakran ■ *Kent State University*

Yang Chen and Dongning Luo ■ *University of North Carolina at Charlotte*

Ye Zhao ■ *Kent State University*

Jing Yang and Wenwen Dou ■ *University of North Carolina at Charlotte*

Shixia Liu ■ *Microsoft Research Asia*

**A**dvanced technologies such as mobile phones and the Internet have greatly increased the quantity and accessibility of text documents. Devices and their users are generating massive numbers of documents at a high frequency—for example, daily, hourly, or by-the-minute emails, messages, and webpages. This has introduced the urgent need for efficient storage, processing, and

analysis of such constantly growing text collections. Recently, researchers have successfully applied visualization tools to process and analyze text data. (For more on this, see the sidebar.)

Visual exploration of text streams is a challenge. First, because these streams continuously evolve, we need visualization aids to trace the temporal evolution of existing topics, monitor emerging topics, and examine the relationships between them. Second, such visualization systems should process text streams

without prescanning an entire stream or assuming a priori knowledge. Third, users should be able to change their information-seeking focus at any

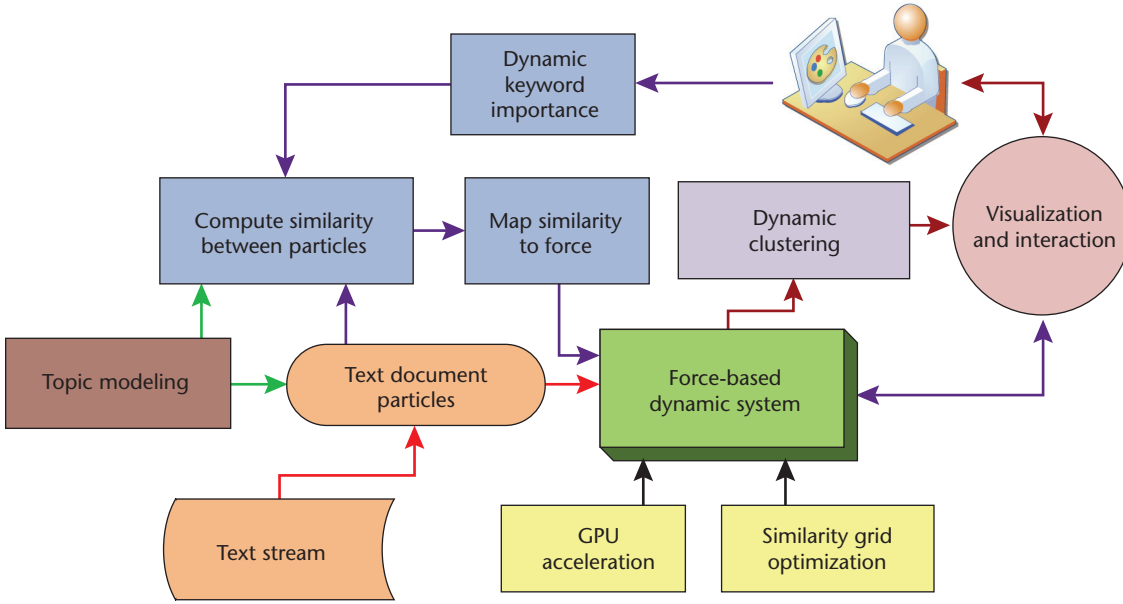
time and receive immediate feedback. Such interactivity is a decisive factor for a visual analytic system in real applications in which domain users usually don't know the text streams in advance. Finally, the system should scale to large volumes of text streams and respond to their evolution in real time.

To meet these challenges, we designed Streamit, a dynamic visualization system for exploring text streams.<sup>1</sup> Figure 1 illustrates Streamit's infrastructure. Streamit is based on a dynamic force-directed simulation into which text documents are continuously inserted. A dynamic 2D display presents the incoming documents. Users can explore documents and document clusters on the basis of keywords or topics. They can discover emerging patterns online by monitoring the real-time display. They can also examine historical data's temporal evolution through animations that play back past streams. Streamit lets users manipulate the display's visual structure on the fly.

Streamit's current version incorporates five features that enable effective text stream visualization. First, users can examine and interact with the stream's continual evolution. Second, Streamit allows dynamic processing through dynamic keyword vectors that upgrade adaptively in accordance with

---

**Streamit lets users explore visualizations of text streams without prior knowledge of the data. It incorporates incoming documents from a continuous source into an existing visualization context with automatic grouping and separation based on document similarities. A powerful user interface allows in-depth data analysis.**



**Figure 1. The Streamit system.** A dynamic 2D display presents continuously incoming text documents. Users can explore the documents and clusters on the basis of keywords or topics in the dynamic display.

incoming documents. Third, *dynamic keyword importance* presents a keyword's importance at a specific time and lets users change the importance and see the results in real time. Fourth, Streamit uses topic modeling and dynamic clustering to increase its scalability. Finally, it employs a similarity grid and parallel processing to optimize performance.

### The Force-Based Dynamic System

Streamit represents each document as a mass particle moving inside a 2D visualization domain. We define the potential energy by pairwise text similarity between documents. Minimizing the system's total potential energy moves similar document particles closer and drives away dissimilar ones, which we achieve by attractive and repulsive forces between particles. Consequently, equilibrium of the particles visually depicts the data clusters and outliers at a particular moment. As new particles are injected, Streamit automatically adjusts its visual output. The dynamic behavior is critical for reducing change blindness when new patterns emerge.

This physical model is well suited for the continuous depiction and analysis of growing document collections. Text documents enter the system at any time and automatically join clusters of related collections. The particles in the system travel continuously under the impact of new particles. So, the visual structures gradually evolve without abrupt changes that break the mental picture users have already formed. Particular particles' erratic motion (for example, moving from one cluster to another) might reveal outliers or significant new trends. This provides an advantage over existing static or time-

window-based visualization approaches, which depict only stationary data patterns or the sporadic transitions between these patterns.

#### Particle Potential

The particles' velocity and acceleration follow Newton's law of motion. Each pair of particles has a potential energy  $\Phi_{ij}$ :

$$\Phi_{ij} = \alpha (|\mathbf{l}_i - \mathbf{l}_j| - l_{ij})^2,$$

where  $\alpha$  is a control constant and  $\mathbf{l}_i$  and  $\mathbf{l}_j$  are the positions of document particles  $p_i$  and  $p_j$ . Whereas  $|\mathbf{l}_i - \mathbf{l}_j|$  represents the two particles' Euclidian distance,  $l_{ij}$  is their ideal distance (we discuss this more in the next section). So, this pair potential function models the deviation of the two particles from their ideal locations, which is achieved at zero potential.

#### Particle Similarity

We determine an optimal layout by defining  $l_{ij}$ . We obtain  $l_{ij}$  from the pairwise similarity computed from the documents' keywords:

$$l_{ij} = 1 - \delta(p_i, p_j),$$

where  $\delta(p_i, p_j) \in [0, 1]$  is the cosine similarity between  $p_i$  and  $p_j$ . Documents with a high similarity will have a smaller  $l_{ij}$  and will be clustered more closely in the visualization.

#### The Force-Directed Model

A global potential function is the sum of the pairwise energy:

## Related Work in Text Visualization

Many text visualization systems use similarity-based projection to help users gain insight from large text collections. James Wise and his colleagues used multidimensional scaling to map documents with similar content closely to each other, thus forming “galaxies” or “mountains” in the displays.<sup>1</sup> Fernando Paulovich and Rosane Minghim proposed a point placement approach to build a hierarchy of documents and project them as circles.<sup>2</sup> Unlike these approaches, Streamit uses a dynamic similarity-based projection system to depict text streams (see the main article).

Related to our aim to handle continuous incoming text streams, TextPool produced a visual summary that clustered related terms as a dynamic textual collage.<sup>3</sup> Unlike Streamit, it visualized very recent stream content as a partially connected graph, which was “not for analyzing any significant portion of stream history.”<sup>3</sup> Also, the graph represented the stream’s salient terms instead of documents.

Pak Chung Wong and his colleagues dynamically visualized stream data in a moving time window using incremental data fusion.<sup>4</sup> Their approach inserted newly arrived data items into the existing layout when the similarity placement’s error was smaller than a given threshold. Once the error exceeded the threshold, their approach recalculated the whole layout. Unlike Streamit, it didn’t address interactive exploration and user control.

Eventriver processed incoming documents on the fly using dynamic keyword processing and an incremental text-clustering algorithm.<sup>5</sup> Individual documents weren’t visible from the overview of the stream. In contrast, Streamit lets users examine individual documents within the global temporal and similarity context.

Elizabeth Hetzler and her colleagues visualized text collections in a 2D projection space, distinguishing fresh and stale documents.<sup>6</sup> They applied In-Spire<sup>1</sup> to dynamic document flow. When new documents were added, their approach adjusted the existing vocabulary content and regenerated the visual results. However, their approach didn’t show an animated transition of the view. In contrast, Streamit reveals the stream’s evolution in detail with controllable transient animations.

Unlike methods that work on static high-dimensional data, our approach visualizes text streams using *force-directed placement* (FDP).<sup>7</sup> FDP has  $O(N^3)$  complexity, which urges researchers to improve its computational performance. On the other hand, improvements restrict force calculations to a subset of the entire data, which could lead to misleading approximated results. To ensure correct dynamic behavior, we avoid reducing the force computation to only some of the particles. Instead, we spatially divide the visualization domain to quickly locate the particles’ appropriate initial position. More important, we use GPU acceleration to fully exploit the parallel nature of the simulation algorithm, which achieves dramatic speedup.

### References

1. J.A. Wise et al., “Visualizing the Non-visual: Spatial Analysis and Interaction with Information for Text Documents,” *Readings in Information Visualization: Using Vision to Think*, Morgan Kaufmann, 1999, pp. 442–450.
2. F. Paulovich and R. Minghim, “Hipp: A Novel Hierarchical Point Placement Strategy and Its Application to the Exploration of Document Collections,” *IEEE Trans. Visualization and Computer Graphics*, vol. 16, no. 8, 2008, pp. 1229–1236.
3. C. Albrecht-Buehler, B. Watson, and D. Shamma, “Visualizing Live Text Streams Using Motion and Temporal Pooling,” *IEEE Computer Graphics and Applications*, vol. 25, no. 3, 2005, pp. 52–59.
4. P.C. Wong et al., “Dynamic Visualization of Transient Data Streams,” *Proc. IEEE Symp. Information Visualization*, IEEE CS Press, 2003, p. 13.
5. D. Luo et al., “Eventriver: An Event-Based Visual Analytics Approach to Exploring Large Text Collections with a Temporal Focus,” to be published in *IEEE Trans. Visualization and Computer Graphics*.
6. E.G. Hetzler et al., “Turning the Bucket of Text into a Pipe,” *Proc. 2005 IEEE Symp. Information Visualization (Infovis 05)*, IEEE CS Press, 2005, p. 12.
7. T. Fruchterman and E. Reingold, “Graph Drawing by Force-Directed Placement,” *Software—Practice and Experience*, vol. 21, no. 11, 1991, pp. 1129–1164.

$$V(\mathbf{l}_1, \dots, \mathbf{l}_N) = \sum_i \sum_{j>1} \Phi_{ij},$$

where  $N$  is the particle number and  $\mathbf{l}_1, \dots, \mathbf{l}_N$  represent these particles’ current locations.

We minimize the system’s potential to an equilibrium state that provides a global optimized placement of these particles. To achieve this optimization, a numerical simulation minimizes the global potential with a sequence of time steps. At each time step, the minimization

leads to forces acting on each particle:

$$\mathbf{F}_i = -\nabla_{\mathbf{l}_i} V(\mathbf{l}_1, \dots, \mathbf{l}_N),$$

which attracts or repulses particles from each other. From Newton’s law, we get

$$\mathbf{F}_i = m_i \mathbf{a}_i,$$

where  $m_i$  is the mass and  $\mathbf{a}_i$  is the particle acceleration. We compute  $\mathbf{a}_i$  as

$$a_i = \frac{2 \sum_j \alpha (|\mathbf{l}_i - \mathbf{l}_j| - l_{ij})}{m_i},$$

which we use to update the location of  $p_i$  at each time step. When every particle no longer moves (in numerical computing, the displacement is smaller than a threshold  $\xi$ ), the system's visual layout is optimized.

Figure 2 describes the basic computing procedure, which assumes that every particle has the same unit mass. The constant  $\alpha$  is an empirical parameter controlling the force ( $\alpha = 0.01$  in our case studies) so that the numerical simulation is stable. That is, no particle will exit the 2D domain or be squeezed to the domain's center.

## Improving Interactive Exploration and Scalability

To provide advanced data exploration, Streamit incorporates the following techniques.

### Dynamic Keyword Importance

Typically, text visualization approaches compute  $\delta(p_i, p_j)$  by a predefined formula—for example, cosine similarity—from the keyword vector of  $p_i$  and  $p_j$ . However, stream text collections usually span a long period of time. For a real-world stream, one keyword might constantly appear over a period of time and then fade out, whereas another might frequently pop up. Although users typically don't have knowledge about the incoming documents, they'll change their focus of interest as the stream evolves. So, the definition and computation of similarity should instead be a function of time and be user-adjustable.

To address this challenge, we propose computing dynamic keyword importance in addition to  $\delta(p_i, p_j)$ , which will let users manipulate the keywords' significance at any time. The classic cosine similarity can be improved as

$$\delta(p_i, p_j) = \frac{\sum_{k=1}^K (w_{ik} I_k)(w_{jk} I_k)}{\sqrt{\sum_{k=1}^K (w_{ik} I_k)^2 \cdot \sum_{k=1}^K (w_{jk} I_k)^2}},$$

where  $I_k$  is the importance of keyword  $k$ ,  $K$  is the number of keywords, and  $w_{ik}$  is the weight of  $k$  in  $p_i$ . The classic cosine similarity can be considered a special case in which  $I_k = 1$ . All the  $K$  weights form this document's keyword vector. Streamit dynamically updates the current vector's length, so that the system can handle live data streams.

We calculate keyword weight as

$$w_{ik} = O_{ik} * \log_2 \frac{N}{n_k},$$

```

Set the maximum displacement  $D$  as a large value
while  $D > \xi$  do
  for  $i = 0$  to  $N - 1$  do
    for  $j = i + 1$  to  $N$  do
       $F_i += 2 * \alpha (|\mathbf{l}_i - \mathbf{l}_j| - l_{ij})$ 
    end for
  end for
  for  $i = 0$  to  $N - 1$  do
     $a_i = F_i / m_i$ 
    update this particle's position
    update the maximum displacement  $D$  of all particles
  end for
end while

```

**Figure 2. The dynamic-simulation algorithm. The force  $F$  is exerted on the particles, causing them to move toward optimal distances. The simulation stops when the displacement is no longer significant. For an explanation of the terms, see the section “The Force-Based Dynamic System” in the main article.**

where  $O_{ik}$  is the occurrence of  $k$  in  $i$ ,  $N$  is the total number of documents, and  $n_k$  is the number of documents in  $N$  that contain  $k$ . The inverse-document-frequency factor  $N/n_k$  favors the keywords concentrated in a few documents of a collection, compared to keywords with a similar frequency that are prevalent throughout the collection.<sup>2</sup>

Users can freely modify the keyword importance through the visual interface, which presents frequent keywords in an ordered list. Furthermore, Streamit can automatically determine importance as follows (we discuss this in more detail later):

$$I_k = a * O_k + b * (te_k - ts_k) + c * n_k. \quad (1)$$

$O_k$  is the occurrence of  $k$  in the current existing documents,  $te_k$  is the last time it appears, and  $ts_k$  is the first time it appears. The equation  $(te_k - ts_k)$  increases the importance for older keywords;  $n_k$  increases the importance for keywords appearing in many different documents. Here,  $a$ ,  $b$ , and  $c$  are positive constants satisfying  $a + b + c = 1$ ; users select them to specify preferences for the three factors. In our experiments, we used  $a = 0.3$ ,  $b = 0.3$ , and  $c = 0.4$ .

### Topic-Based Visualization

Streamit might have trouble revealing clusters when the keyword space's dimensionality is too high, owing to the lack of data separation in such high-dimensional spaces. Recent topic-modeling techniques, such as latent Dirichlet allocation (LDA),<sup>3</sup> reduce the keyword-document space to a much lower feature space that not only is intuitive to interpret but also captures most of the variance in the corpus. In particular, topic modeling automatically represents the documents using a



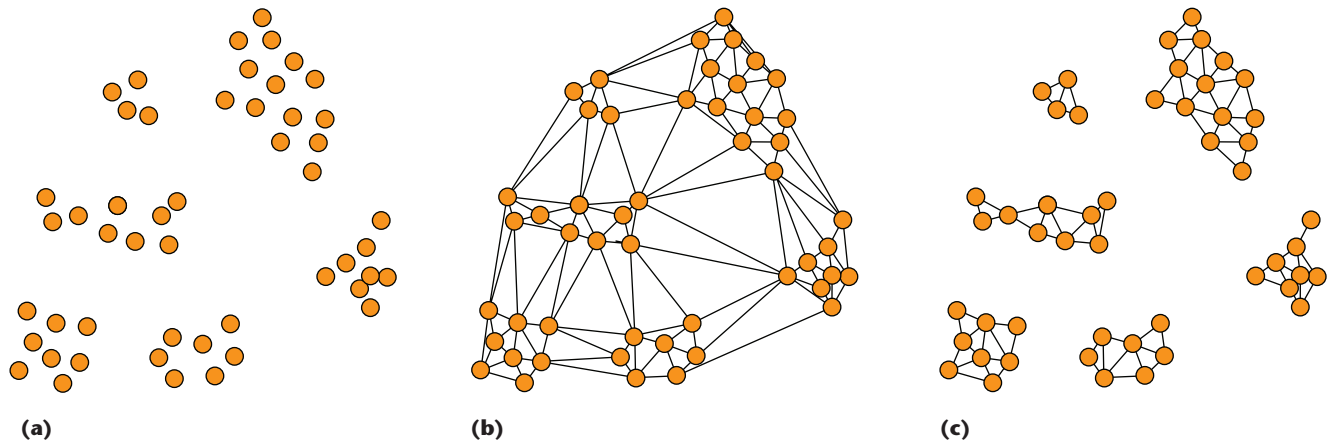


Figure 3. Cluster generation. (a) The original particles. (b) Delaunay triangulation results. (c) Graph cut results. The graph cuts partition the particles into disjoint sets (that is, connected components) representing the semantic clusters.

set of probabilistic topics, which are described by a probability distribution over keywords.

We use LDA to extract topics from a large document archive that's highly related to the text stream to be visualized, such as its historical archives. We associate each extracted topic with a set of keywords highly related to the stream. During the ongoing visualization, Streamit dynamically examines whether an incoming document contains the extracted topics according to its keywords. It then represents the document by a vector of the probable weights of that document's topics. All the aforementioned calculation, visualization, and interactions based on keywords can be applied to the topics.

The benefits are significant. Because there are many fewer topics than keywords, the documents are clustered better. Because the topics are at a higher semantic level than the keywords, users can more easily understand the generated clusters (we discuss this in more detail later). For example, interdisciplinary proposals covering multiple topics, which are difficult to identify in the keyword-based approach, can be easily detected in the topic-based approach.

### Dynamic Clustering

The distribution of document particles in the 2D space lets users visually identify clusters of documents with similar semantics. However, on the basis of individual particles, it's difficult to conduct cluster-level operations, such as selecting all documents in a cluster and examining a cluster's semantics. To address this problem, Streamit automatically discovers clusters from the evolving geometric layouts, so that they can be explicitly presented and manipulated through the visual interface. Moreover, the visualization can display a text stream at the cluster level to reduce clutter and enhance scalability.

**Cluster generation.** At a particular moment, a group of document particles can be considered to form a semantic cluster with this definition:

If particles  $s$  and  $t$  are in a cluster, there must at least exist a path between  $s$  and  $t$  that connects a sequence of particles,  $s, p_0, p_1, \dots, p_c, t$ , with a pairwise line segment,  $s \rightarrow p_0, p_0 \rightarrow p_1, \dots, p_c \rightarrow t$ . The connected segments' maximum length is shorter than the predefined threshold  $\zeta$ .

Here we use the single-linkage rule to define clusters, which considers connected components (with respect to  $\zeta$ ) as one cluster. We discover such clusters directly from the 2D geometric layout. In particular, we can apply a typical agglomerative algorithm to partition all particles into clusters.

Starting with  $N$  particles forming  $N$  clusters, we repeatedly merge two clusters according to the distance between their nearest neighbors. This straightforward approach has  $O(N^2)$  complexity and doesn't use the particles' geometric layout. However, it represents the resultant clusters only by individual particles and provides no topological information.

Because an effective visualization should distinctly show these clusters' spatial areas, we apply a computational-geometry method to create a simple polygon from each cluster's particles. Similarly to research in spatial data mining,<sup>4</sup> we propose a two-step algorithm (see Figure 3):

1. Apply Delaunay triangulation for all particles in the system.
2. In the created graph (the triangle mesh), cut the edges that are longer than  $\zeta$ .

The graph cuts partition the particles into disjoint sets (that is, connected components) representing the semantic clusters.

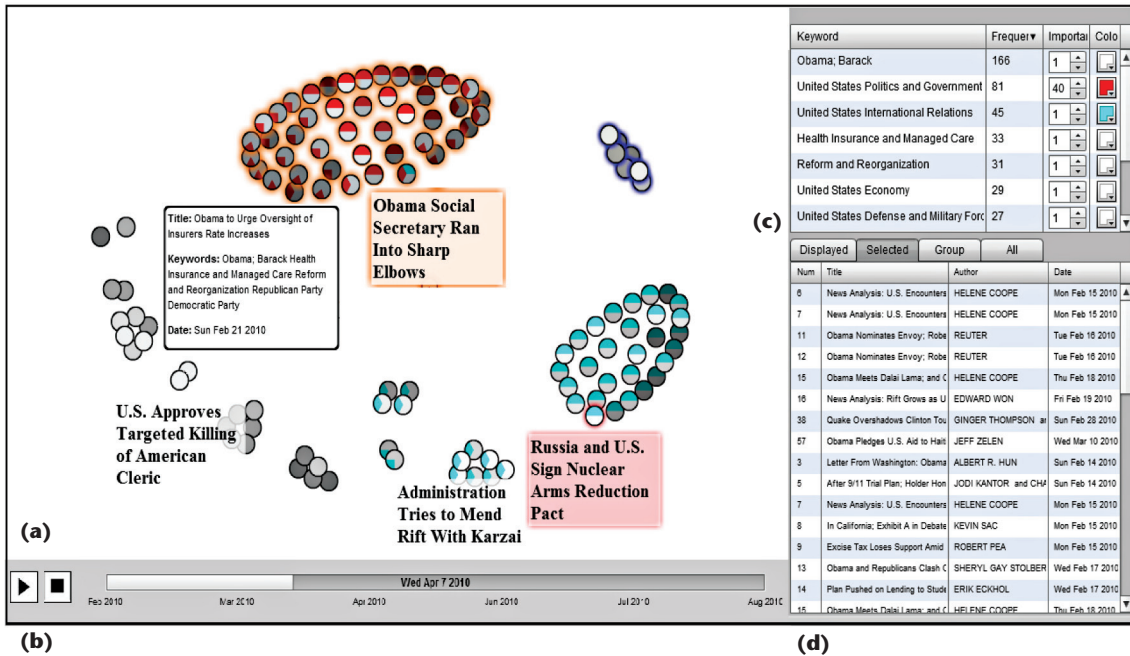


Figure 4. Streamit's interface. (a) The main window displays the particles' movement. (b) The animation control panel lets users navigate through the simulation. The (c) keyword table and (d) document table are synchronized with the particle stream to maintain an up-to-date list of the keywords and documents.

This method has a complexity of  $O(N \log N) + O(E)$  for  $N$  particles, with  $O(N \log N)$  for Delaunay triangulation and  $O(E)$  for browsing all  $E$  edges in the graph cuts. In Delaunay triangulation, the maximal number of edges is  $3N - 6$ ,  $O(E) \sim O(N)$ . So, our method achieves  $O(N \log N)$ , which is computationally faster than the agglomerative algorithm.

**Cluster evolution.** The created clusters merge and split over time when new documents arrive or keyword importance changes. Such evolution is critical for knowledge discovery and should be tracked and visualized.

To track cluster evolution, we give each cluster a distinct ID; each particle carries its cluster's ID. To manage cluster identification in a context-aware way, we compute a preferred ID list for each new cluster. The list ranks the IDs of all the particles in the cluster before the update, according to the number of the particles with the same ID. The largest new cluster receives the top ID in its list. Then, we iteratively choose a cluster according to its size (that is, number of particles). Each cluster receives an ID following the order of its preferred list, if we haven't assigned that ID to other clusters. If all ID choices are occupied, a cluster receives a new ID that didn't appear in the previous step. So, large clusters tend to keep their contextual information over time.

We associate each ID with a color, which we also assign to the cluster with that ID. To avoid color clutter, we set a threshold  $K$  of the number of significant clusters. The largest  $K$  clusters appear

with background halos in their colors; the other clusters don't. (We discuss these things more in the next section.)

## Visualization

Streamit has a main window, an animation control panel, a keyword or topic table, and a set of document tables (see Figure 4). Refer to the video at <http://doi.ieeecomputersociety.org/10.1109/MCG.2011.91> for a description of the visual layout and a few examples on real datasets.

### The Main Window

This window (see Figure 4a) visually presents the particles' movement through an animated 2D display. A pie chart represents each particle, with colored slices indicating keywords of interest. The pies' closeness reflects the documents' similarities. The pies' positions dynamically change to reveal the stream's temporal evolution. Shades of gray indicate document age; the older a document is, the darker it is. Users can adjust the pies' size and transparency to reduce clutter that appears as the number of documents grows.

### The Animation Control Panel

Streamit buffers recent documents into a moving time window called the *buffer window*, which is larger than the window displaying the current documents. Users can play back the animation in the buffer window to examine the buffered stream's temporal and semantic evolution in detail.

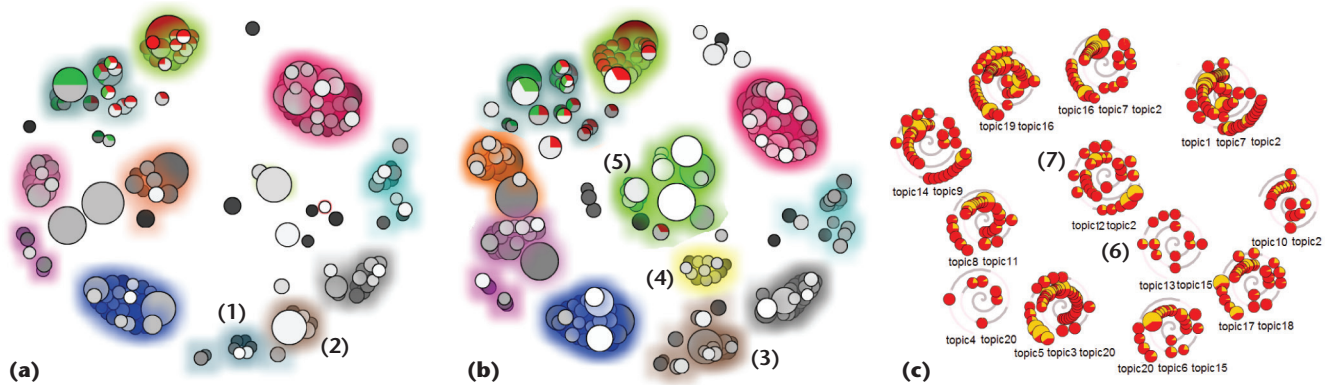


Figure 5. US National Science Foundation Information and Intelligence Systems (NSF IIS) award data with topic modeling. (a) On 1 Sept. 2000, there were 172 research projects. Haloes indicate significant clusters. Red pie slices indicate topic 16 in Table 1; green slices indicate topic 19. (b) On 15 Sept. 2001, there were 330 research projects. Clusters 1 and 2 from Figure 5a have merged into cluster 3. Clusters 4 (mainly about topic 13) and 5 (mainly about topic 12) are new. (c) A spiral view of Figure 5b lets users examine the clusters' temporal trends. Clusters 6 and 7 indicate spiral views of clusters 4 and 5, respectively, in Figure 5b.

Table 1. A table of topics.

Topic no.	Keywords
2	Data, Mine, Cluster, Graph, Biology, Analysis, Discovery
6	Image, Scene, Model, Recognition, Language, Shape, Speech
12	Biological, Protein, Genome, Search, Gene, Sequence, Patent
13	Video, Motion, ASL, 3D, Camera, Sign, Dance
15	Image, Speech, Haptic, Display, Impair, Auditory, Graphic
16	Query, Database, Data, XML, Stream, Edu
19	Data, Workflow, Privacy, Management, Web, Metadata

The animation control panel controls playback (see Figure 4b). Users can move the slider to start the animation at any point in time. They can also pause the display to examine a moment of the stream or change parameters such as keyword importance.

#### The Keyword or Topic Table

Streamit provides keyword information in a table that's updated dynamically (see Figure 4c). This table lists the keywords for the displayed documents and their frequencies, importance, and colors.

Alternatively, the table can display topic information. In this case, it shows not only the topics but also their significant keywords.

#### Document Tables

Users can click on a tab to view one of four tables (see Figure 4d):

- buffered documents,
- documents displayed in the main window,
- documents selected by users, and
- document clusters generated by the system or created by users.

Users can sort the documents by their authors or time stamps. They can also click on a title to access the document's full text.

#### Labeling

Because document titles contain rich semantic information in a condensed manner, Streamit uses them as document labels. However, displaying all the titles could create severe clutter. So, we developed a labeling algorithm to provide the most recent semantic information with user-controllable clutter levels.

Streamit generates clusters according to a dissimilarity threshold. In each cluster, the dissimilarities among the documents are less than the threshold, and only the most recently arrived document is labeled. By changing this threshold, users can control the label clutter. A newly arrived document is either assigned to an existing cluster or forms a new cluster. So, no label will change except that of the affected cluster. This feature maintains temporal consistency among adjacent displays. The newest injected document will always be labeled, which is usually desired in text stream visualization.

Figure 4a shows the automatic-labeling results. Streamit highlighted the newest injected document and its label in red and highlighted the selected documents and their labels in orange.

Labels and particles might overlap when many documents are displayed. Streamit displays labels on the top of particles and lets users interactively change the transparency of the labels' backgrounds. An opaque background makes labels easy to read; a semitransparent background lets users examine particles underneath the labels. Users can turn all labels off or turn a specific label on or off by clicking on it.

#### The Visual Representation of Clusters

After Streamit creates clusters, triangle meshes represent their outlines. Streamit places background halos in the mesh area for significant clusters (see Figures 5a and 5b).



Users can also examine clusters' temporal trends in a less cluttered spiral view.<sup>5</sup> Figure 5c shows a spiral view of 12 clusters. Each spiral is a time axis located at the center of a cluster in the original 2D display. The cluster documents, displayed as pie charts, are mapped to the spiral according to their time stamps. The pie charts' distribution indicates the cluster's temporal trends.

In the topic-modeling view, pies have two colors indicating how documents are related to the cluster's general topics. In Figure 5c, red indicates the number of topics the document shares with other documents in the cluster. Yellow indicates the number of other topics in this document.

Users can quickly examine an unknown cluster through a keyword cloud triggered by selecting the cluster. It displays the cluster's most significant semantic information—the titles of the most recently arrived documents and the keywords with the highest TF-IDF (term frequency-inverse document frequency) weights.

Figure 6 shows the keyword clouds for clusters 4 and 5 in Figure 5b. The keywords appear below the titles; a keyword's size indicates its weight. Users can interactively set the colors of the background, titles, and keywords.

## Interaction

Streamit lets users interactively manipulate the visualization according to varying interests. They can also search, track, and examine documents.

### 2D Display Manipulation

Streamit provides four types of 2D display manipulation.

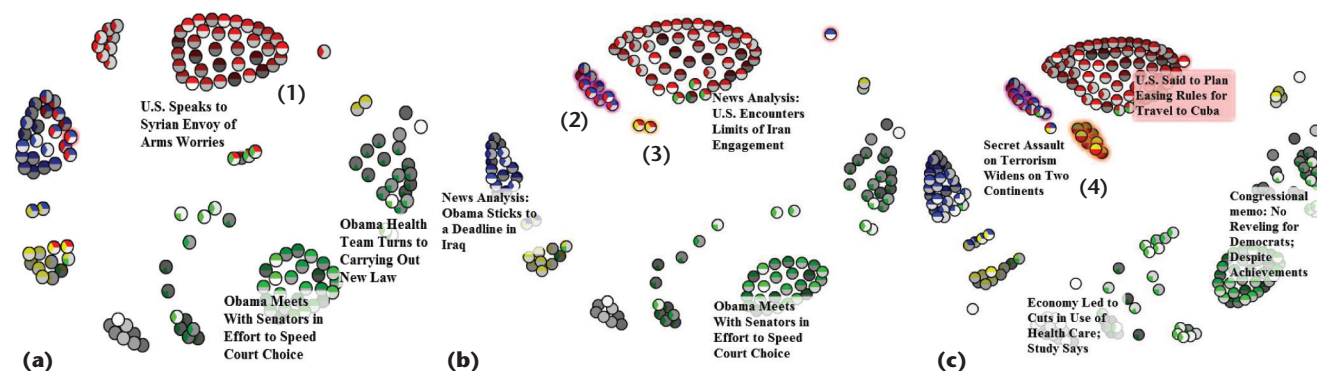


Figure 7. Barack Obama news. (a) On 13 Aug. 2010, there were 136 news articles. Cluster 1 indicates a group of "US and International Affairs" articles. (b) After we increased the importance of the keyword "International Relations," Streamit placed articles with this keyword more closely together. Cluster 2 is a group of articles we labeled "War"; cluster 3 is the "Terrorism" group. (c) On 18 Sept. 2010, there were 230 news articles. Cluster 3 in Figure 7b has grown considerably, and a recent news article (cluster 4) now comes between it and cluster 2. Green indicates "Politics," red indicates "International Relations," yellow indicates "Terrorism," and blue indicates "Defense and Military."

Augmenting the Periphery with Larg...  
MLIAM: NESPOLE! - Negotiating through Spoken L...  
information video 3d vision activities visual  
image design motion images learning virtual object web  
scene asl robot displays display sequence clustering

(a)

Collaborative Research: ASES: An ...  
ITR: An Active; Personalized; Adaptive; Multi-form...  
biological sequence protein data genomic images  
retrieval queries design molecular edu learning query

(b)

Figure 6. Keyword clouds for clusters (a) 4 and (b) 5 in Figure 5b. The keywords appear below the titles; a keyword's size indicates its weight. By clicking on a keyword, users can select all documents with this keyword from the cluster.

**Adjusting keyword importance.** Users can adjust the keyword importance in the keyword table to emphasize specific content and receive an immediate response (see Figure 7).

**Grouping and tracking documents.** Users can use the different-colored halos to highlight selected groups or automatically computed clusters, which promotes easy document tracking in the dynamic display. Figure 4a shows a cluster in orange and one in blue.

**Browsing and tracking keywords.** Users can assign colors to keywords of interest to track them. In



each pie, a colored slice's size is proportional to its keyword's weight in the document. Users can investigate keyword and document relations and track the evolution of relevant topics in this way (see Figure 7). They can also click on a keyword of interest in the keyword table, and halos will highlight all documents with that keyword. Users can sweep the keyword table in this way to find keywords of interest.

---

## ***Users can choose the current selected documents as examples and select documents that are within a specified distance from them.***

---

**Setting moving windows.** Users can interactively change the length of the moving window—that is, the period to be investigated—of the currently displayed documents.

### **Document Selection**

Streamit offers five ways to select documents.

**Manual selection.** Users can manually select documents from the document tables or use rubber-band selection by dragging the mouse. Halos highlight the selected documents (see Figure 4a), and the selected document table displays the documents' information (see Figure 4d).

**Example-based selection.** Users can choose the current selected documents as examples and select documents that are within a specified distance from them. They can easily control the range to select similar documents.

**Keyword-based selection.** Users can select multiple keywords from the keyword table (see Figure 4c). Streamit then automatically selects and highlights the related documents (see Figure 4a).

**Cluster-based selection.** Users can click the background halo of a cluster or its spiral to select all the documents in it. They can also click a keyword in a keyword cloud to select all the documents in that cluster that contain the keyword.

**The shoebox.** Users might want to focus on temporal evolution and examine the selected documents later. They can easily send those documents into the shoebox for later examination of the full text.

## **Three Case Studies**

For these case studies, we first sorted documents in prerecorded collections by their time stamps. We then fed them to Streamit at intervals of a few seconds to simulate a quickly evolving text stream.

### **Exploring Barack Obama News**

We explored a text stream of 230 *New York Times* news items about Barack Obama reported between 19 July and 18 September 2010. We gave the keywords the tags that come with the news. In each document, we assigned the occurrences of the keywords a value of one. The buffer window covered the whole stream. Streamit automatically assigned keyword importance, using Equation 1.

Figure 7a shows the display for 13 August 2010, which represents 136 news articles. On that date, keywords such as “Politics and Government,” “International Relations,” “Defense and Military,” and “Terrorism” had high-frequency values, according to the keyword table. We assigned them distinct colors to track the articles characterized by them, as Figure 7a illustrates.

We then increased the importance of “International Relations.” Figure 7b shows that the articles with this keyword are closer than in Figure 7a. We easily selected them using rubber-band selection and found in the shoebox that they contained keywords such as “China,” “Terrorism,” and “Afghanistan War.”

We wanted to focus on “Afghanistan War” and “Terrorism” because most of those news articles had been recently inserted (and therefore had a lighter shade of gray). We clicked on “Afghanistan War” to select the related articles and created a group named “War” for them. We highlighted that group with pink halos (see cluster 2 in Figure 7b). We created another group for “Terrorism” in the same way and highlighted it with orange halos (see cluster 3 in Figure 7b).

Then, we continued to play the animation and track these groups' evolution. Figure 7c shows the visualization when all the news articles were displayed. Cluster 3 in Figure 7b grew considerably. Also, a recent news article (cluster 4 in Figure 7c) came between it and cluster 2 in Figure 7b. This new cluster was related to both “Afghanistan War” and “Terrorism.” We selected this article and read it in full detail by clicking on the pie.

### **Exploring NSF Award Abstracts**

We explored 1,000 US National Science Foundation Information and Intelligent Systems (NSF IIS) award abstracts funded between March 2000 and August 2003 as a text stream. The time-varying

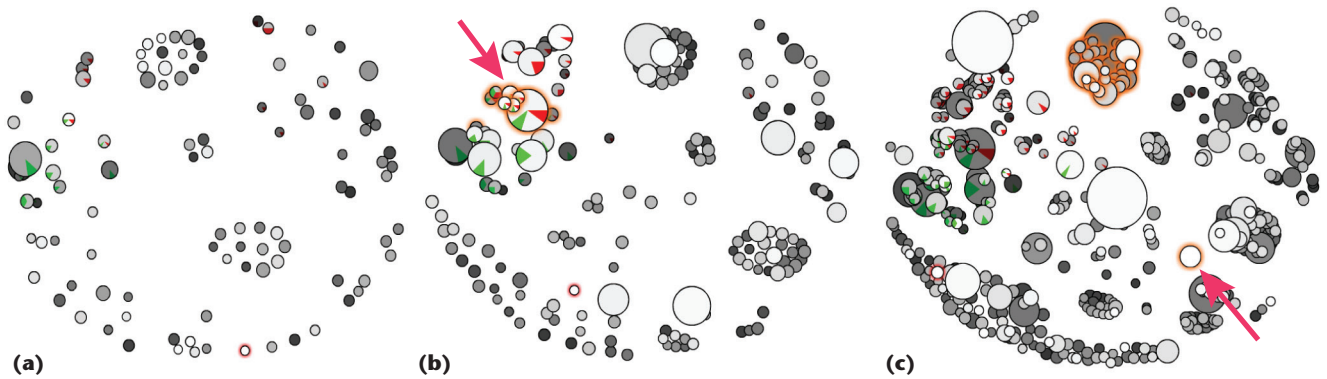


Figure 8. NSF IIS award collections. (a) On 1 Aug. 2000, there were 95 research projects. The arrow indicates abstracts containing the keywords “Management” (green pie slices) and “Database” (red slices). (b) On 1 Sept. 2000, there were 172 research projects. We increased the two keywords’ importance so that we could observe the relevant abstracts more easily. (c) On 15 Mar. 2002, there were 672 research projects. The arrow indicates a potential transformative proposal. When the stream further evolved, we observed that IIS continuously supported projects with the two keywords.

funding behavior is critical in understanding research and administrative trends. We automatically characterized each document with a set of keywords; its corresponding pie’s size was proportional to the project’s funding.

Figures 8a and 8b show the stream in two adjacent months. Multiple large projects started in the second month. We paused the animation, selected items of interest, and examined them in detail. From the shoebox, we observed that the keywords “Management” and “Database” appeared in many of these project’s abstracts. We highlighted “Management” in red and “Database” in green. We also increased their importance so that we could observe the relevant abstracts more easily (see Figure 8b). Although some abstracts contained both keywords (indicated by the arrow in Figure 8b), many others contained only one of them. We pulled back the animation to the previous month (see Figure 8a) to examine these topics’ temporal evolution. When the stream further evolved, we observed that IIS continuously supported projects with these keywords (see Figure 8c).

In Figure 8c, we highlighted all projects containing “sensor” with halos. The node with a halo indicated by the arrow is a potential transformative proposal because it’s far from the other projects with halos. We examined this abstract in detail and found that it was a project about just-in-time information retrieval on wearable computers.

### Evaluating Topic Modeling and Dynamic Clustering

Using the NSF data from the previous case study, this case study applied topic modeling and dynamic clustering to increase Streamit’s scalability.

Figures 8b and 5a illustrate keyword-based and topic-based visualization (respectively) for the same set of documents for the same month. Many documents that seemed unrelated to other docu-

ments in Figure 8b actually belonged to clusters in the higher semantic level in Figure 5a. Table 1 lists topics involved in Figure 5a. Topics 16 (the red slices in Figure 5a) and 19 (the green slices in Figure 5a) contain “Database” (the red slices in Figure 8b) and “Management” (the green slices in Figure 8b). The related clusters’ semantics are easier to understand in the topic-based visualization than in the keyword-based visualization.

Figures 5a and 5b demonstrate dynamic cluster evolution. Figure 5a shows a handful of clusters. Cluster 1 is mainly about topic 15, and cluster 2 is mainly about topic 6. In Figure 5b, these two clusters merge into one larger cluster (cluster 3). Two new clusters also appear: cluster 4 is mainly about topic 13, and cluster 5 is mainly about topic 12. The visual effects help users identify critical data variations. Figure 6 shows the two new clusters’ keyword clouds.

Figure 5c displays the spiral view of the clusters in Figure 5b. Cluster 6 represents cluster 4 in Figure 5b, and cluster 7 represents cluster 5 in Figure 5b. The two or three most significant topic names for a cluster appear below its spiral. Each pie’s red slice indicates its document’s relationship to the cluster’s main theme. Large pies (projects with funding of over US\$1 million) typically have a small red slice, indicating that they’re probably interdisciplinary projects. In contrast, small projects usually involve fewer topics and agree more with the clusters.

### Performance Optimization

Here we show how we applied parallel computing and a similarity grid to improve Streamit’s scalability for large datasets.

#### GPU Acceleration

Our computational algorithm is inherently parallel at each simulation step. So, we accelerate the

**Table 2. Streamit's performance on a CPU and GPU for some *New York Times* text streams.**

Document time period	No. of documents	No. of keywords	Avg. simulation time per frame (ms)		GPU/CPU speedup	Max. simulation time (ms)		Avg. no of simulation steps per frame
			CPU	GPU		CPU	GPU	
13 Feb.–18 Aug. 2010	6,157	5,057	540	34	17.9	4,350	230	173
1 Aug.–31 Oct. 2006	7,100	1,059	620	41	15.1	9,070	480	177
1 July–31 Aug. 2010	10,205	2,036	986	53	15.9	11,030	610	200
Synthetic dataset	15,000	2,000	1,020	65	15.7	13,070	682	196

**Table 3. Dynamic-clustering performance.**

No. of documents	Avg. time per frame (ms)		Max. time per frame (ms)	
	CPU	GPU	CPU	GPU
6,157	1,270	317	2,875	334
7,100	1,546	324	3,484	334
10,205	2,544	330	5,625	341
15,000	4,247	332	9,423	356

**Table 4. The performance optimization obtained by employing similarity grids on a dataset of 7,100 documents.**

Similarity grid size	Avg. no. of simulation steps
No grid	225
20 × 20	207
50 × 50	177
100 × 100	182
200 × 200	186

computation on graphics hardware with a CUDA (Compute Unified Device Architecture) implementation similar to an  $N$ -body problem.<sup>6</sup>  $N$  particles execute their force-placement algorithm simultaneously as individual threads distributed to a grid of CUDA blocks. Each thread accesses and updates the particle's position from the information of the last step loaded into the blocks' shared memory.

We conducted experiments using *New York Times* text streams. We ran the experiments (using a 50 × 50 similarity grid) on both an Nvidia NVS 295 GPU with 2 Gbytes of memory and a 1.8-GHz Intel Core 2 CPU with 2 Gbytes of RAM. For each frame, the simulation ran multiple steps with  $\xi$  preset at  $10^{-4}$ .

As Table 2 shows, the GPU performed much better than the CPU. On average, GPU acceleration achieved real-time performance of 25 to 30 fps, over 15 times faster than the CPU version. The maximum simulation time after each document insertion on the GPU was less than a second. This was sufficiently fast, considering the relatively slower response time for human perception and analysis of the visualization update.

We also tested Streamit on a synthetic dataset with approximately 15,000 documents and 2,000 keywords. The results (see the last row of Table 2)

showed that our system worked well for such a large text stream on the GPU.

Table 3 reports the performance of triangulation-based dynamic clustering. With GPU acceleration, the cluster generation doesn't impose much extra overhead on the system.

### Applying a Similarity Grid

Document particles' initial positions significantly affect the computational cost. We employ a similarity grid to ensure that Streamit inserts new documents near similar documents. The grid divides the 2D visualization domain into rectangular cells with a given resolution. Each cell actively maintains a special keyword vector consisting of the average keyword weights computed from the cell's documents. For a new document, we first compute its vector's similarity to the other cells' vectors to find the most similar one. We then place this document at that cell's center. An appropriate resolution will provide good acceleration and won't be too large because of extra overhead.


Table 4 shows the average number of simulation steps for 7,100 documents at different grid sizes. A 50 × 50 grid decreased the simulation steps per frame to 78 percent of the steps needed without the grid. The execution time decreased at the same ratio.

### Discussion

The performance optimization makes our system applicable for monitoring live streams. The *New York Times* news feed is produced continuously, averaging three documents per hour and achieving a maximum of eight documents per hour at peak times. The minimum interval between documents is approximately one minute.

A capable real-time visualization system should handle newly inserted items faster than this minimum interval. From Table 2, the maximum simulation time on the CPU is a few seconds. With GPU acceleration, the handling time decreases to less than one second. So, our system can effectively handle this news stream, displaying many thousands of documents for analysis. Note that an ordinary consumer PC and graphics card provide this capability.

We're upgrading our system to handle even larger text streams with advanced GPUs. It's important not to overwhelm the users with the flood of information. Our system lets users manipulate the simulation speed, and they can pause the system and save clusters or documents for further investigation. Future improvements will also exploit unbalanced text-streaming speed, which occurs when texts excessively arrive during some events but rarely arrive under ordinary situations.

**W**e'll integrate Streamit with online topic-modeling techniques and deploy it on the Internet to visualize text streams with frequently evolving topics—for example, Twitter. We'll also conduct user studies to further assess the system's feasibility. 

### Acknowledgments

US National Science Foundation grants IIS-0915528, IIS-0916131, IIS-0946400, and NSF DACS10P1309 and the Ohio Board of Regents partly supported this research. We thank the anonymous reviewers for helpful reviews and Zhi Yuan for improving the system.

### References

1. J. Alsakran et al., "Streamit: Dynamic Visualization and Interactive Exploration of Text Streams," *Proc. 2011 IEEE Pacific Visualization Symp. (PacificVis 11)*, IEEE Press, 2011, pp. 131–138.
2. G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management*, vol. 24, no. 5, 1988, pp. 513–523.
3. D.M. Blei, A.Y. Ng, and M.I. Jordan, "Latent Dirichlet Allocation," *J. Machine Learning Research*, Jan. 2003, pp. 993–1022.
4. I.-S. Kang, T.-W. Kim, and K.-J. Li, "A Spatial Data Mining Method by Delaunay Triangulation," *Proc. 5th ACM Int'l Workshop Advances in Geographic Information Systems (GIS 97)*, ACM Press, 1997, pp. 35–39.
5. J. Carlis and J. Konstan, "Interactive Visualization of Serial Periodic Data," *Proc. 11th Ann. ACM Symp. User Interface Software and Technology*, ACM Press, 1998, pp. 29–38.
6. L. Nyland, M. Harris, and J. Prins, "Fast N-Body Simulation with CUDA," *CPU Gems 3*, H. Nguyen, ed., Addison-Wesley Professional, 2007, pp. 677–696; [http://developer.nvidia.com/GPUGems3/gpugems3\\_ch31.html](http://developer.nvidia.com/GPUGems3/gpugems3_ch31.html).

**Jamal Alsakran** is a PhD candidate in Kent State University's Department of Computer Science. His research focuses on multidimensional and text visualization, and visual analytics. Alsakran has an MS in computer science from the University of Jordan. Contact him at [jalsakra@cs.kent.edu](mailto:jalsakra@cs.kent.edu).

**Yang Chen** is a PhD student in the Department of Computer Science at the University of North Carolina at Charlotte. His research interests include visual analytics and information visualization. Chen has a BS in science from Wuhan University. Contact him at [ychen61@uncc.edu](mailto:ychen61@uncc.edu).

**Dongning Luo** is a PhD student in the Department of Computer Science at the University of North Carolina at Charlotte. His research interest is information visualization. Luo has a BS in information engineering from Shanghai Jiao Tong University. Contact him at [dluo2@uncc.edu](mailto:dluo2@uncc.edu).

**Ye Zhao** is an assistant professor in Kent State University's Department of Computer Science. His research interests include natural-phenomena modeling, data visualization, and visual analytics. Zhao has a PhD in computer science from Stony Brook University. Contact him at [zhao@cs.kent.edu](mailto:zhao@cs.kent.edu).

**Jing Yang** is an associate professor in the Computer Science Department at the University of North Carolina at Charlotte. Her research interests include visual analytics and information visualization. Yang has a PhD in computer science from Worcester Polytechnic Institute. Contact her at [jing.yang@uncc.edu](mailto:jing.yang@uncc.edu).

**Wenwen Dou** is a PhD candidate in computer science at the University of North Carolina at Charlotte. Her research interests are visual analytics and human-computer interaction. Dou has a BS in telecommunications engineering from the Beijing University of Posts and Telecommunications. Contact her at [wdou1@uncc.edu](mailto:wdou1@uncc.edu).

**Shixia Liu** is a lead researcher at Microsoft Research Asia. Her research interests include visual text analytics and visual social-network analysis. Liu has a PhD in computer graphics and computer-aided design from Tsinghua University. Contact her at [shliu@microsoft.com](mailto:shliu@microsoft.com).



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

Visit CG&A on  
the Web at  
**[www.computer.org/cga](http://www.computer.org/cga)**

