

AniViz: A Template-Based Animation Tool for Volume Visualization

Hiroshi Akiba, Chaoli Wang, and Kwan-Liu Ma ■ *University of California, Davis*

Visualization is a tool for both exploration and communication. Whereas interactive visualization is the key to insightful exploration, animation can effectively convey a complex process or structure. In particular, animation provides a powerful means for illustrating objects' evolution and interaction in a complex environment. It lets us observe how an object changes its shape, size, position, and spatial relationship to other objects over time.

Despite the extensive use of animation to effectively present information, existing data visualization software systems provide limited support for making animations. Some more helpful systems let users specify keyframes and then interpolate between the keyframes to create a sequence. For example, Anima is a keyframe-based system that produces animation sequences for scientific visualizations.¹ It combines a simulation with light effects, camera settings, and geometric transformations.

However, even with such support, nonexpert users, such as application scientists, might have difficulty specifying and managing keyframes. Deriving a sequence of desired keyframes is a painstaking process entailing much trial and error. This issue becomes more critical for visualizing time-varying, multivariate volume data, because scientists must consider a larger parameter space. Surprisingly, researchers haven't done much to improve this situation.

To help alleviate this problem, we developed AniViz—a tool for making visualization animations of time-varying, multivariate volume data. With it,

users can create an animation sequence by defining instances from motion templates and then manipulating and combining those instances. When integrated into a visualization system, AniViz lets scientists create an animation while visualizing and exploring their data. This is different from creating an animation with a postprocessing tool using a set of pregenerated keyframes, which significantly limits what scientists can create. The freedom to create and blend different effects by operating on different instances encourages expressiveness, leading to animation results that scientists couldn't easily create with existing visualization systems (see the “Animation Support in Visualization Systems” sidebar).

From Computer Animation to AniViz

Our research is inspired by well-established computer animation R&D and increasing demands from scientific collaborators for better animation support. In computer animation, you can perform motion synthesis by defining motion primitives, such as walk, run, or jump, and then assembling animation frames from those primitives.² Moreover, a significant advance in animation is nonlinear animation (NLA). Offered by many packages such as Autodesk 3ds Max, Autodesk Maya, and Softimage, NLA lets animators create animation beyond keyframes by letting them

AniViz is an animation tool following two principles. First, it's desirable to directly turn the results of data exploration and visualization into animation content. Second, users can create a complex animation sequence by combining several simple effects. Such effects, and operators to combine them, are fine-tuned via an intuitive user interface.

Animation Support in Visualization Systems

Most visualization systems provide some animation support. For example, VisIt adopts the keyframe approach.¹ Users record a session by storing keyframe parameters into files; they then create movies from session files by specifying movie parameters through a GUI or command-line interface. This approach, however, doesn't allow intuitive specification of animation content at the semantic level.

ParaView also uses the keyframe approach; it provides an animation panel with VCR-like buttons for animation playback.² ParaView's track editor lets users easily create, delete, or adjust individual keyframes. This function is similar to many video-editing software tools and our AniViz interface (see the main article).

AVS/Express (Application Visualization System/Express) lets users develop a keyframe animation using an animator module.³ Users can set the properties of module parameters to determine which parameters of the view are interpolated to create the animation.

EnSight provides basic functions to export rendered images to an animation.⁴ It includes EnVe, a general-purpose animation postprocessing tool that splices a series of images or animation files to generate a new animation. EnSight also uses keyframe animations. Users can control the movement

through or around the model and can control what entities the animation shows. In addition, EnSight supports complex keyframe operations such as spline camera control.

A major difference between these visualization systems and AniViz is the expressive power it offers. Users can create animations not only through keyframes but also by using templates, instances, and operators to specify semantic content during data exploration and visualization.

References

1. H. Childs et al., "A Contract Based System for Large Data Visualization," *Proc. 2005 IEEE Conf. Visualization (VIS 05)*, IEEE CS Press, 2005, pp. 191–198.
2. C.C. Law, A. Henderson, and J. Ahrens, "An Application Architecture for Large Data Visualization: A Case Study," *Proc. IEEE Symp. Parallel and Large-Data Visualization and Graphics*, IEEE CS Press, 2001, pp. 125–128.
3. H.D. Lord, "Improving the Application Development Process with Modular Visualization Environments," *ACM Siggraph Computer Graphics*, vol. 29, no. 2, 1995, pp. 10–12.
4. "EnSight Tutorials," EnSight, 2010; www.ensight.com/tutorials.html.

combine, mix, and blend motions to produce entirely new animations.

Analogously, we define primitives (which we call templates) and operations to create animations for

scientific visualization. The templates cover animation techniques commonly used in scientific visualization. We derived them by examining animations typically made for research presentations, science exhibitions, and instructions.^{3–8} For instance, using our technique, you can easily create a video like the one NASA scientists created for Tropical Storm Edouard⁹ by combining templates for spatial exploration and transfer-function exploration with an operator for blending parameter spaces.

AniViz Basics

In AniViz, we classify a list of motion types; each type corresponds to a template, which consists of a set of visualization parameters. Once the user has specified the parameter values, a template becomes an instance representing a piece of visualization content. (The relation between template and instance is analogous to that of class and object in object-oriented programming.) We chose this design because we prefer an intuitive way to specify animation content but also want a flexible, extensible approach that lets users easily construct and customize their animations. To connect instances, AniViz provides a set of operators.

Templates and Instances

Table 1 shows some volume visualization templates. The overview template gives users an over-

Table 1. Some templates for creating animations to visualize time-varying, multivariate volume data.

Template	Description
Overview	Animate through a certain set of visualization parameters (spatial, temporal, variable, or transfer function) with a default path or scheme.
Spatial exploration	Interpolate the camera parameter between camera parameters c_1 and c_2 , using an interpolation function f_c .
Temporal exploration	Animate time steps between ts_1 and ts_2 with a playback function f_t .
Variable exploration	Change a variable from v_1 to v_2 with a blending function f_v .
Transfer-function exploration	Interpolate the transfer function between tf_1 and tf_2 with an interpolation function f_{tf} .
Highlighting	Change the transfer function tf , the lighting parameter l , or the camera parameter c of objects of interest with a periodic function f_p .
Slicing	Change a slicer's position and orientation with texture tex mapped onto it.
Image	Blend an instance with an image frame to provide the context.
Caption/annotation	Add captions before an instance, or add annotations on certain frames of an instance.

Animation-Enhanced Visualization

Visualizations frequently use animation. The simplest form is to play back a time sequence. A more interesting form is to use animation to enrich visualization or highlight information. For example, Eric Lum and his colleagues presented *kinetic visualization*, which adds animated particles flowing over the surface of an object to enhance the perception of static objects' shapes and spatial relationships.¹ Carlos Correa and Deborah Silver produced animations that highlight features or enhance a particular aspect of a dataset by traversing the volume along a specified path via a transfer function.² Jonathan Woodring and Han-Wei Shen incorporated animation into visualization to highlight objects in volume data.³ As highlighting mechanisms, they used positional motion and opacity variation.

Ivan Viola and his colleagues presented a solution for automatic focusing on features in a volumetric dataset.⁴ The user selects a focus from a set of predefined features, and the algorithm automatically determines the most expressive view of the features. This solution estimates characteristic viewpoints using mutual information from information theory. The camera path changes smoothly by switching the focus from one feature to another.

Konrad Mühler and his colleagues introduced viewpoint-selection techniques to enhance animation cre-

ation and interactive exploration in medical-intervention planning.⁵ To guide viewpoint selection, they use a list of parameters such as the unoccluded surface's size, the occluding objects' importance, the preferred region, and viewpoint stability.

References

1. E.B. Lum, A. Stompel, and K.-L. Ma, "Kinetic Visualization: A Technique for Illustrating 3D Shape and Structure," *Proc. 2002 IEEE Conf. Visualization (VIS 2002)*, IEEE CS Press, 2002, pp. 435–442.
2. C.D. Correa and D. Silver, "Dataset Traversal with Motion-Controlled Transfer Functions," *Proc. 2005 IEEE Conf. Visualization (VIS 05)*, IEEE CS Press, 2005, pp. 359–366.
3. J. Woodring and H.-W. Shen, "Incorporating Highlighting Animations into Static Visualizations," *Proc. 2007 IS&T/SPIE Electronic Imaging Conf.*, SPIE, 2007, article 649503.
4. I. Viola et al., "Importance-Driven Focus of Attention," *IEEE Trans. Visualization and Computer Graphics*, vol. 12, no. 5, 2006, pp. 933–940.
5. K. Mühler et al., "Viewpoint Selection for Intervention Planning," *Proc. Joint Eurographics/IEEE VGTC Symp. Visualization*, Eurographics, 2007, pp. 267–274.

all impression of the data. The spatial-exploration, temporal-exploration, variable-exploration, and transfer-function-exploration templates let users customize the parameter sets. The highlighting template is useful for multivariate-data visualization in which users animate one variable's opacity and simultaneously observe that variable's relationships with other variables. The slicing template helps users observe internal structures by cutting through the volume. Users can superimpose the image template with other instances to display the context. Finally, with the caption/annotation template, users can provide transitions between different pieces of animation content or explain the data.

This template-based design isn't limited to those example templates. It's flexible enough to incorporate state-of-the-art animation-based visualization schemes (see the "Animation-Enhanced Visualization" sidebar).

Users can create multiple instances from one common template, each having a different set of parameter values. They can also opt to directly specify keyframes in the traditional way. From the keyframe and animation context, AniViz can automatically infer what types of templates and instances these keyframes correspond to. Users can edit instances through the user interface, which we describe later.

Operators

AniViz uses three types of operators.

Parameter-space blending. This operator creates intermediate frames between two instances I_1 and I_2 by interpolating their respective parameters. If I_1 and I_2 don't overlap along the time axis, we generate intermediate frames by interpolating the parameters of the last frame of I_1 and the first frame of I_2 . Otherwise, we generate intermediate frames by interpolating the parameters of their corresponding frames.

Figure 1 illustrates parameter-space blending with a computed-tomography head data set. This operator reveals the content buried between the sets of parameters for two instances, thus generating novel image frames. When we apply this operator to transfer-function-related instances, we can interpolate transfer functions in several ways. For a 1D transfer function, we can treat the color and opacity transfer functions as four arrays of RGBA components each and linearly interpolate the two sets of four arrays. Or, we can make a correspondence between one segment in a transfer function and another segment in another transfer function, and interpolate both their positions and values. The examples in Figure 1 use this scheme. For example, in Figure 1b, the transfer function

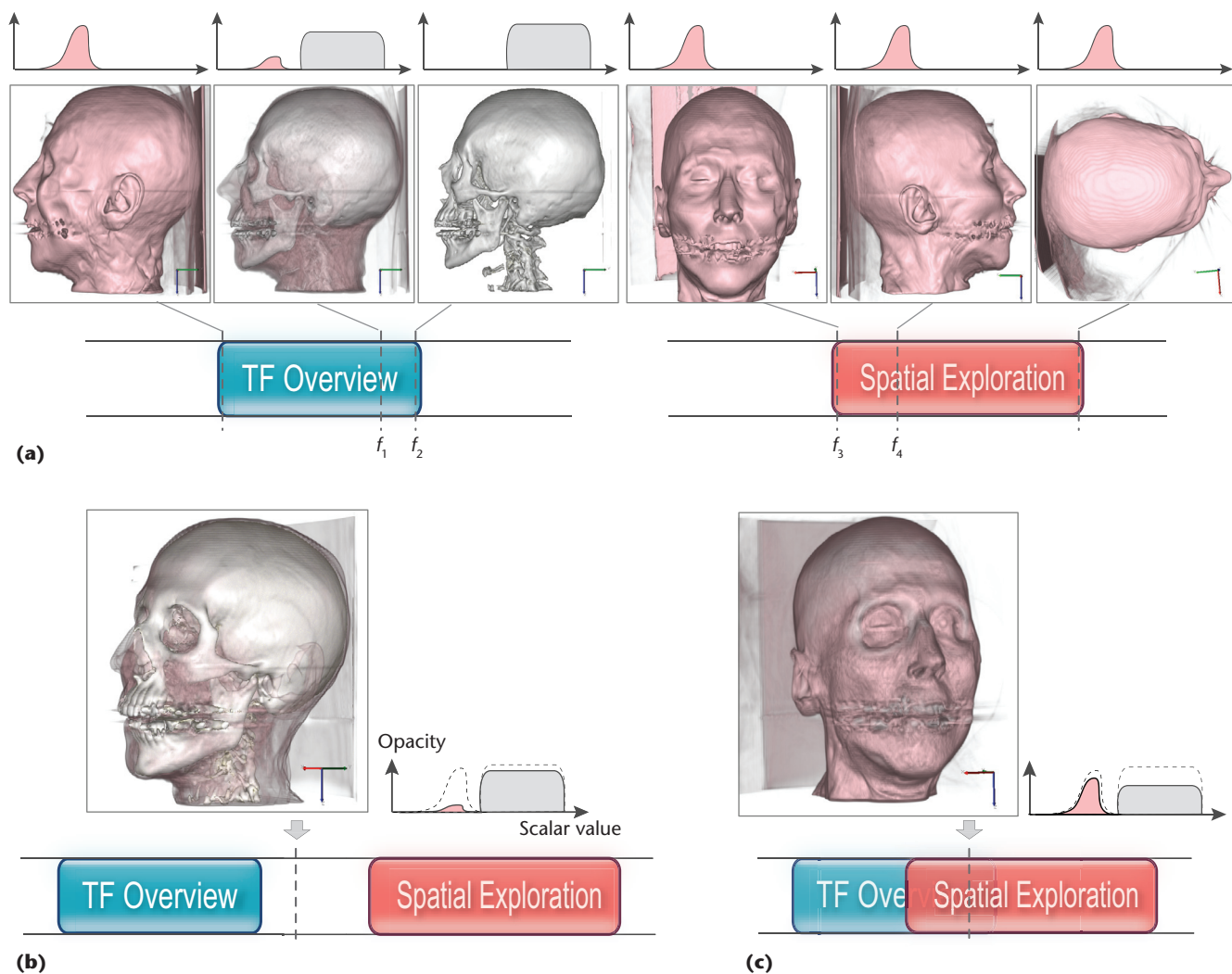


Figure 1. Two cases of parameter-space blending. (a) Selected frames of transfer-function-exploration and spatial-exploration instances. (b) In case 1, the two instances don't overlap; we generate the intermediate frame by interpolating the parameters of frames f_2 and f_3 . (c) In case 2, the two instances overlap; we generate the intermediate frame by interpolating the parameters of f_1 and f_4 .

content from frame f_2 gradually fades out as the transfer function content from frame f_3 gradually fades in. Some special templates that aren't derived from the visualization content (such as the caption or image/annotation template) can't be used with this operator.

Image-space blending. This operator creates the animation content between I_1 and I_2 by interpolating their respective image frames. Similarly to parameter-space blending, if I_1 and I_2 don't overlap along the time axis, we generate intermediate frames by blending the last frame of I_1 and the first frame of I_2 . The effect is that the last frame of I_1 gradually fades out as the first frame of I_2 gradually fades in. If I_1 and I_2 overlap, we generate intermediate frames by blending their corresponding frames.

Figures 2 and 3 illustrate image-space blending with combustion and solar-plume datasets. This

operator is typically used with two instances having no clear correspondence. It's useful when two frames from each instance carry important features that shouldn't be altered by changing their respective parameters. This operator's typical uses are for fade-out and fade-in effects and for observing relationships by superimposing frames. The input could come from two frames of different variables (see Figure 2), two frames of different time steps (see Figure 3), or one caption and one rendered frame. You must pay attention when using this operator and interpreting its results because image-space blending doesn't reflect an actual rendering directly from the data.

Playback. This operator lets users repeatedly loop through one or more consecutive instances of interest. This is useful when users want to show certain motions or focus on something. The user specifies the number of loops and the playback

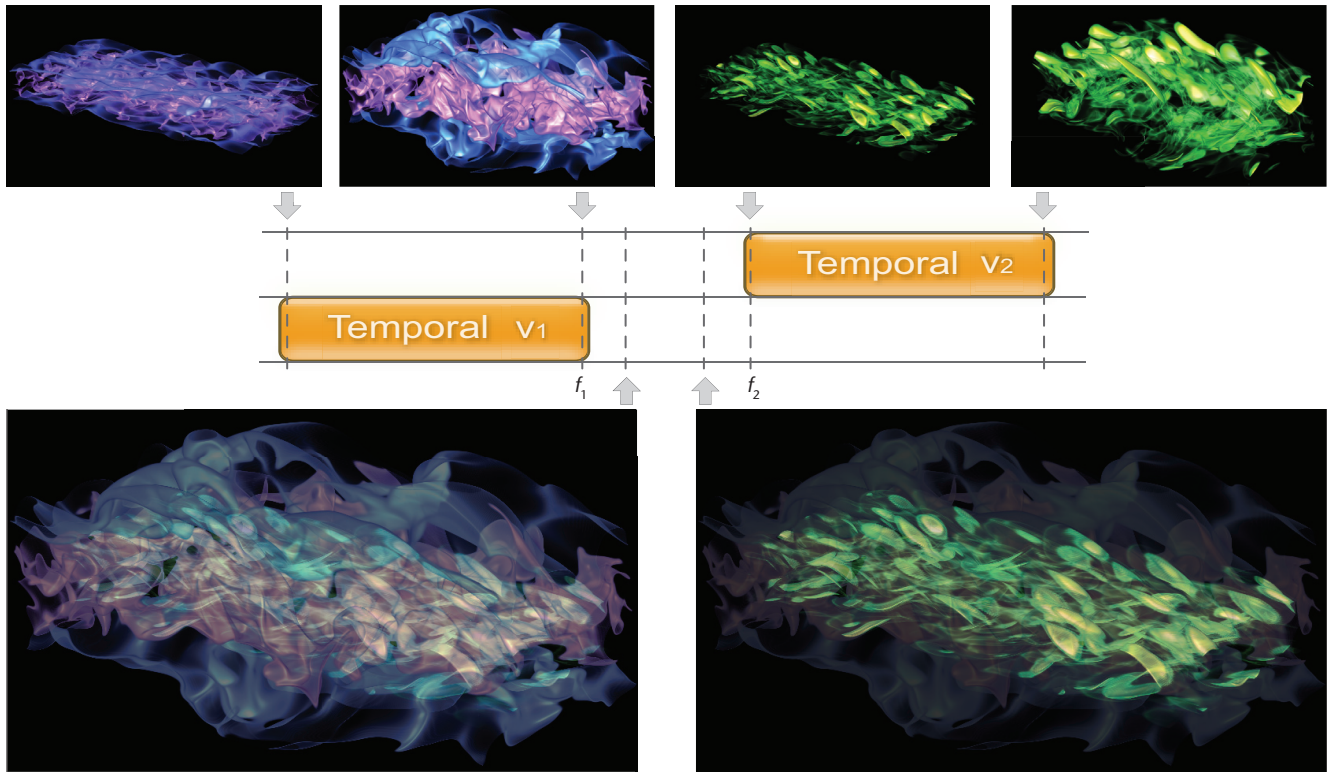


Figure 2. Using image-space blending for fade-out and fade-in effects. The resulting two frames (on the bottom) are generated by overlaying frames f_1 and f_2 in the image space. We determine each intermediate frame's opacity weight by its distances to f_1 and f_2 .

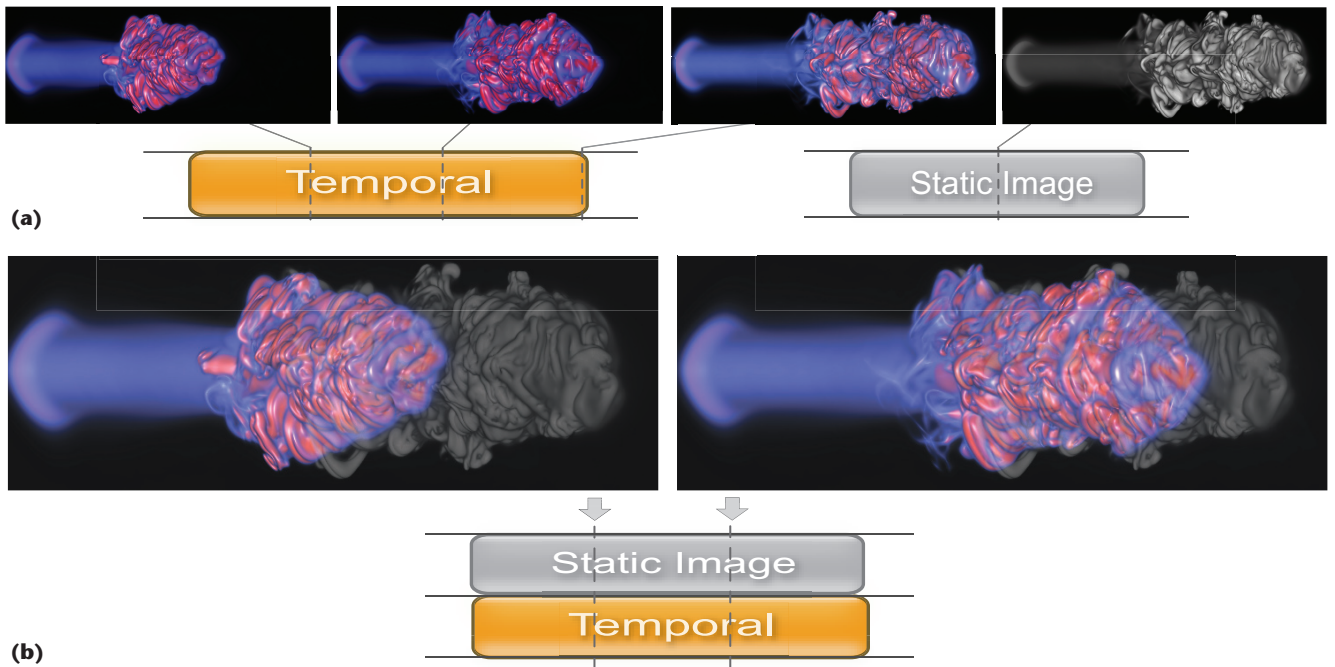


Figure 3. Using image-space blending to examine frames at different time steps. (a) Selected frames of temporal-exploration and image instances. (b) The results of blending. The two instances overlap; we generate intermediate frames by blending their corresponding frames. The image instance shows the frame at the last time step, which provides a reference when blended with the temporal-exploration instance.

type. Playback types include forward, backward, a combination of both, or more sophisticated non-linear playback. This operator has no restrictions; users can apply it to any instance.

The User Interface and Interaction

Figure 4 shows the AniViz interface, which has three main components. The *explorer* lets users interactively explore data sets and define instance

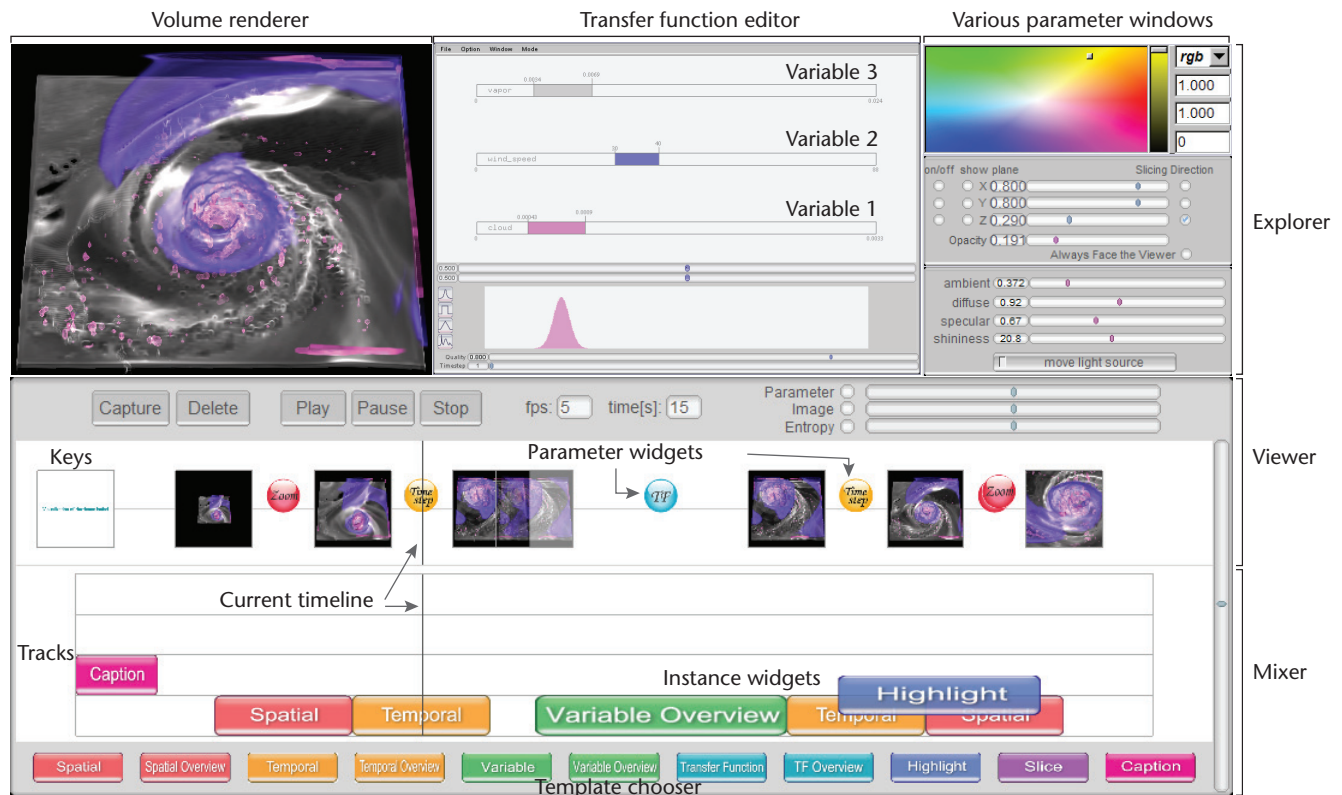


Figure 4. The AniViz interface comprises the explorer (top), including a volume renderer, a transfer function editor, and various parameter specification windows; the viewer (middle); and the mixer (bottom), including multiple tracks and a template chooser.

parameters. The *mixer* lets users assemble multiple instances and derive an animation's overall flow. The *viewer* gives a preview of the final animation output and lets users tune instance parameters.

The mixer and viewer resemble the common interface found in general animation or video-editing software and tools such as Windows Movie Maker, TechSmith Camtasia Studio, and Ulead VideoStudio. We chose this familiar interface style so that users can quickly learn to use AniViz by leveraging their experience with other tools that share a similar layout.

The Explorer

This component (see the top of Figure 4) consists of a volume renderer, a transfer-function editor, and other parameter-specification windows. The transfer-function editor is a simplified trispace interface¹⁰ that lets users explore the data in spatial, temporal, and variable domains in a guided fashion. The explorer includes a time-step slider and widgets for setting other parameters. The explorer presented here is just one design example; it can be augmented or replaced with other designs if necessary.

The Mixer

This component (see the bottom of Figure 4) includes a template chooser and multiple *tracks*

stacked along the vertical axis. The horizontal axis indicates the actual animation time. An instance appears as a colored rectangle, which we call an *instance widget*. Different-colored instance widgets indicate different template types. Users can place an instance widget along any track in the mixer by dragging and dropping a template from the template chooser. They can then resize the widgets, reposition them, or remove them from the tracks. The widgets' positions explicitly indicate their order in the final animation.

More important, as Figures 1, 2, and 3 illustrate, the instance widgets' positions also imply how the instances are combined. If two widgets are in the same track or are offset by half of the track's height along the vertical axis, the mixer uses parameter-space blending. (The offset avoids occlusion because instance widgets could have overlapped each other entirely; see the highlighting instance in Figure 4.) If two instance widgets are in different tracks, the mixer uses image-space blending.

The Viewer

As the user organizes instance widgets in the mixer, the viewer (see the middle of Figure 4) summarizes the animation result. The viewer consists of *keys* (thumbnail images) and their corresponding *parameter widgets*, along with an animation time indicator. Each key shows an actual frame

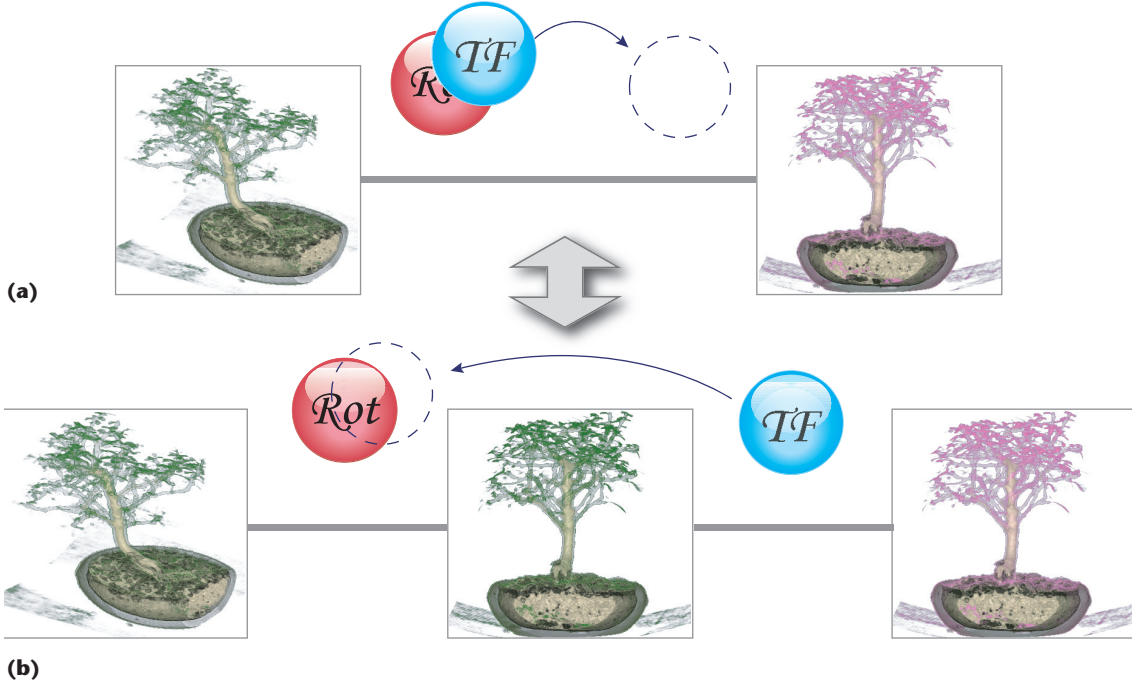


Figure 5. Drag-and-drop interaction with parameter widgets to (a) insert a new key or (b) merge two keys, illustrated with a bonsai data set. “Rot” means “rotate” and “TF” means “transfer function.” This interaction associated with expanding and collapsing visual feedback lets the user easily fine-tune the animation parameters.

of the final animation. The keys are selected either automatically whenever the motion type changes or manually, when the user explicitly sets them. The viewer serves not only as a preview for the animation but also as an editor for individual instances. The user can modify instances’ parameter values anytime while creating an animation.

Parameter widgets show the change of parameters between two consecutive keys. If more than one parameter changed between two keys, their parameter widgets are stacked together. A parameter widget is spherical with a name label. Similarly to instance widgets, different-colored parameter widgets indicate different parameter types.

Using intuitive drag-and-drop interaction, users can insert a new key or merge two keys (see Figure 5). To create a key, users separate a stack of parameter widgets into two or more groups. To merge keys, they stack one parameter widget back on top of another.

Animation Time Allocation

Each animation has a certain time budget. We allocate time to each piece of animation content according to its importance or complexity. A key’s position in the viewer indicates when it appears in the actual animation sequence. To adjust the duration between keys, users can manually move a key by dragging it along the horizontal axis.

Alternatively, AniViz can automatically allocate animation time for all the keys. By allocating more

time to complex scenes and less to uniform scenes, AniViz can present information within the budget and avoid abrupt changes or flickering in the animation.

To quantify the information that changes between two keys, we use the following three distance measures.

The *image distance* is the summation of pixel-wise differences between two consecutive keys:

$$d_1(i_1, i_2) = \sum_{j=1}^M |\bar{u}_1(j) - \bar{u}_2(j)|,$$

where $\bar{u}_1(j)$ and $\bar{u}_2(j)$ are the CIELUV color vectors at the j th pixel in images i_1 and i_2 , respectively. M is the number of pixels in the image. $|\bar{x}|$ denotes the L^2 norm of \bar{x} . We use the CIELUV space because the distance calculated in it roughly approximates the perceived color difference.¹¹

The *parameter distance* is the Euclidean distance calculated in the high-dimensional parameter space, with each parameter carrying a user-specified weight:

$$d_2(\bar{p}_1, \bar{p}_2, \bar{w}) = \sqrt{w_1(p_{11} - p_{21})^2 + \dots + w_n(p_{1n} - p_{2n})^2},$$

where $0 < w_1, w_2, \dots, w_n < 1$ and $w_1 + w_2 + \dots + w_n = 1$. \bar{p}_1 and \bar{p}_2 are the parameter vectors associated with the two keys. n is the number of dimensions in both vectors. w_i is the weight for the parameter pair (p_{1i}, p_{2i}) , $1 \leq i \leq n$.

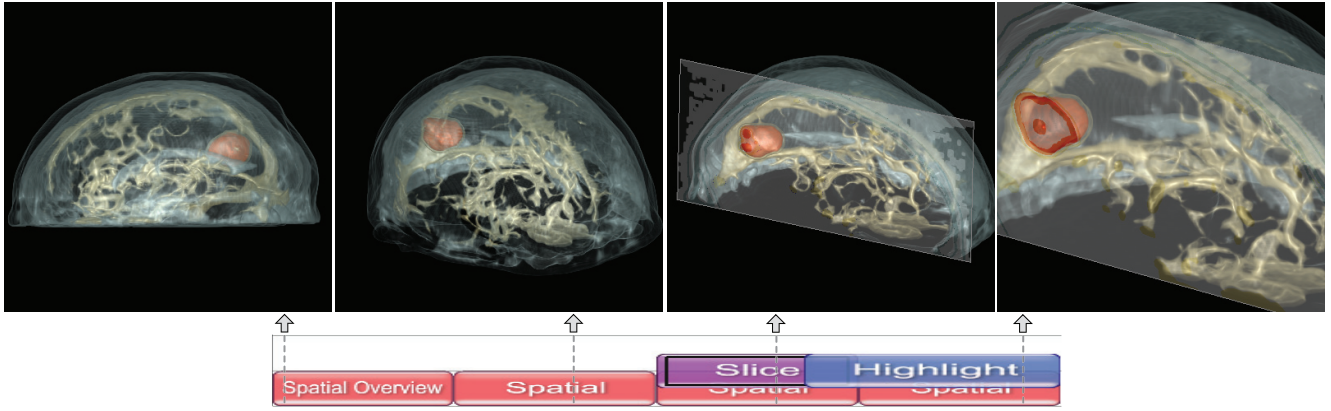


Figure 6. Highlighting a tumor (rendered in red) in the MRI head data set. This case study used the slicing and highlighting templates. The vertical timeline indicates the time point for each of the four snapshot frames in the animation.

The third measure is the *Kullback-Leibler (KL) distance*. Let h_1 and h_2 denote the probability density functions of pixel values in the two keys. We calculate the KL distance (also called the *relative entropy*) as

$$d_3(h_1 \parallel h_2) = \sum_{i=1}^B h_1(i) \log \frac{h_1(i)}{h_2(i)},$$

where $h_1(i)$ and $h_2(i)$ are the normalized heights of the i th histogram bin, and B is the number of bins in the histogram of the keys.

The inputs to each of the three measures are quite different. The image distance takes the actual pixel values, the parameter distance takes the parameters for deriving the image frames, and the KL distance takes the image histograms.

Each measure has its own advantages and disadvantages. For example, the KL distance captures the overall change of two image histograms but is rotation invariant (when the rotation axis is perpendicular to the image plane). However, we can use the image distance to capture the difference introduced by rotation.

So, a better way to use these three measures is to take a hybrid distance heuristically as the weighted sum of the three distances:

$$d = c_1 d_1 + c_2 d_2 + c_3 d_3, \quad (1)$$

where $0 \leq c_1, c_2, c_3 \leq 1$ and $c_1 + c_2 + c_3 = 1$. The animation time allocated between keys is proportional to their hybrid distances. In this way, our solution ensures that the animation content is evenly distributed over the animation sequence.

Case Studies

We demonstrated AniViz with three case studies: one with an MRI head dataset, one with a time-varying, multivariate hurricane dataset, and one with a time-varying combustion dataset. Our system used a GPU-accelerated volume renderer to

enable interactive data exploration. We performed all three studies on a desktop PC with a 3.2-GHz Intel Pentium 4 processor, 2 Gbytes of main memory, and an Nvidia GeForce 7900 GTX graphics card with 512 Mbytes of video memory.

MRI Head Data

This case study (see Figure 6) focused on highlighting a brain tumor. The animation comprised four pieces of animation content. The first was a spatial overview (usually the first step for exploring data) that rotated the volume data 360 degrees along the y-axis. The second piece was a spatial exploration in which the user customized the view. The third was a parameter-space blending between a spatial exploration and a slicing, which revealed the tumor's inner structure. The user played back this piece twice to repeatedly observe the spatial relationship between the tumor and head. The last piece was a parameter-space blending between a spatial exploration and a highlighting, which highlighted the tumor by varying the opacity while zooming in on the region of interest.

To specify the types of playback and number of loops for a group of instances, the user right-clicked on the group and selected the types and the number from a popup menu. The spatial overview, three spatial explorations, and slicing instance came from direct keyframe specification in the traditional way. The highlighting instance was generated from the template chooser.

Hurricane Data

In this case study (see Figure 7), the animation comprised five pieces of content. The first was a caption showing the animation's content, blended with a spatial exploration that zoomed in on the data. The second piece was a temporal exploration to show early time steps. The third was a variable overview that browsed through three variables: vapor, wind speed, and cloud. This overview let the

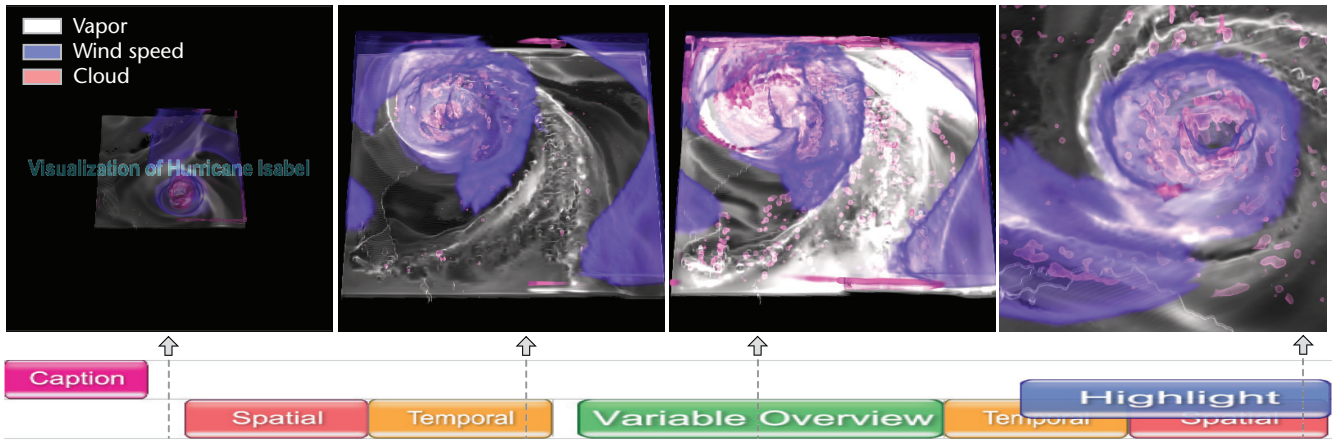


Figure 7. Exploration in the temporal and variable domains of the hurricane dataset. This case study examined three variables: vapor, wind speed, and cloud. It used both parameter-space blending and image-space blending.

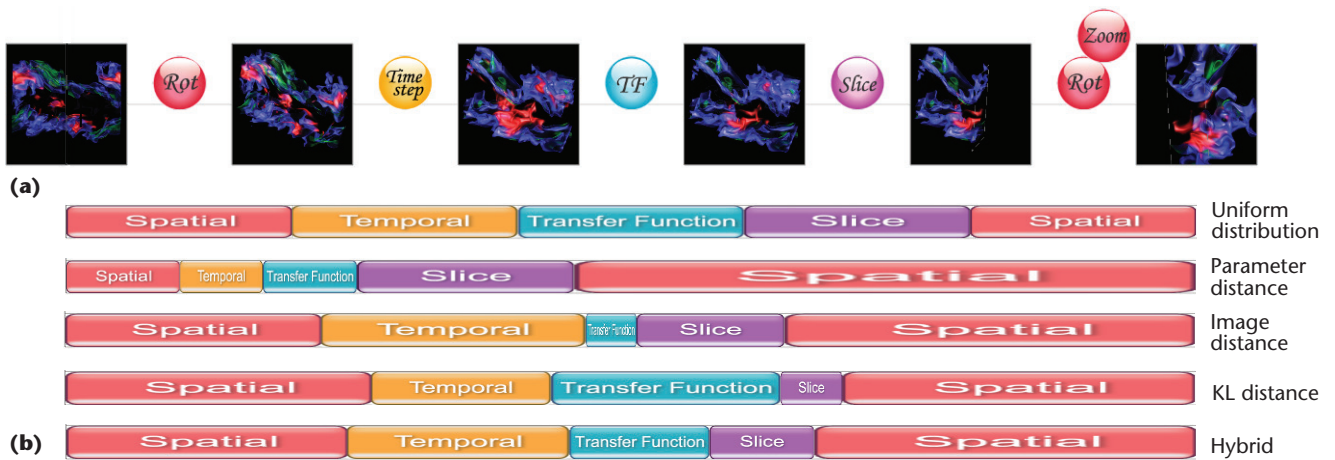


Figure 8. Animation time allocation for the combustion dataset. (a) The uniform time allocation result with the corresponding viewer display, for reference. (b) The instance widgets resized on the basis of allocated time for the parameter, image, and hybrid distances. The hybrid solution yielded the best result because it more evenly distributed the information throughout the animation.

user clearly examine each variable in detail, while maintaining each variable's context. The fourth piece was a temporal exploration to show later time steps; the fifth was a spatial exploration that zoomed in on the hurricane's eye. Furthermore, the user blended the temporal and spatial explorations with a highlighting of the cloud. Creating such a complex animation, which normally might prove difficult without hard-coding all the parameters, is easy with AniViz.

Time-Varying Combustion Data

This case study examined animation time allocation schemes based on the parameter, image, and KL distance measures. As Figure 8 shows, the parameter distance allocated much of the time to the rightmost spatial instance, in which two parameters (rotation and zoom) changed simultaneously. This wasn't desirable because other instances received a relatively short animation time.

The results improved when we used the image

distance and KL distance because these measures take into account information content from the rendered images. However, the image distance and KL distance allocate a very short time for the transfer function and slicing instances, respectively, which could also be undesirable.

The hybrid solution yielded the best result because, as we mentioned before, it more evenly distributes the information throughout the animation.

Discussion and Future Work

AniViz is valuable for adding animation to volume visualization, but we could enhance it in several ways to make it even more useful.

AniViz's object-oriented design makes it easily extensible. Besides volume visualization, AniViz could work with other rendering algorithms, such as surface rendering or geometric-model rendering. For these new applications, we'll evaluate other appropriate animation functionalities and add them as templates.

Conceptual Models and Visual Media

Researchers have explored new models and media to effectively convey information through visualization. Stephen Wehrend and Clayton Lewis proposed a taxonomy of visualization techniques that helps users design applications according to specific visualization goals.¹ Issei Fujishiro and his colleagues introduced Gadget (Goal-Oriented Application Design Guidance for Modular Visualization Environments), an enhanced modular visualization environment.² Gadget lets the user design visualization networks on the basis of the methodologies for classifying visualization techniques and for showing visualization examples.

For time-varying-data visualization, Gareth Daniel and Min Chen introduced *video visualization*. This technique extracts meaningful information from original video datasets and conveys the information to users through a novel visual representation.³

Michael Wohlfart and Helwig Hauser introduced the concept of volume visualization as storytelling.⁴ Their volumetric storytelling process has two phases (story authoring and storytelling) that give users greater control of data presentation. Aidong Lu and Han-Wei Shen presented an interactive-storyboard approach to observing overall data content and changes in time-varying volume data.⁵

By quantifying data differences for automatic selection of representative time steps, their approach summarizes a time-varying dataset in an automatic layout with snapshot images and paths connecting them to indicate data relationships.

References

1. S. Wehrend and C. Lewis, "A Problem-Oriented Classification of Visualization Techniques," *Proc. 1990 IEEE Conf. Visualization (VIS 90)*, IEEE CS Press, 1990, pp. 139–143.
2. I. Fujishiro et al., "Gadget: Goal-Oriented Application Design Guidance for Modular Visualization Environments," *Proc. 1997 IEEE Conf. Visualization (VIS 97)*, IEEE CS Press, 1997, pp. 245–252.
3. G. Daniel and M. Chen, "Video Visualization," *Proc. 2003 IEEE Conf. Visualization (VIS 03)*, IEEE Press, 2003, pp. 409–416.
4. M. Wohlfart and H. Hauser, "Story Telling for Presentation in Volume Visualization," *Proc. Joint Eurographics/IEEE VGTC Symp. Visualization*, Eurographics, 2007, pp. 91–98.
5. A. Lu and H.-W. Shen, "Interactive Storyboard for Overall Time-Varying Data Visualization," *Proc. IEEE Pacific Visualization Symp.*, IEEE CS Press, 2008, pp. 143–150.

AniViz should also let users define their own templates. For example, if a user determined that combining temporal exploration and highlighting was effective and useful, he or she could define a new template including these two and use it directly. Such flexibility would let users familiar with AniViz customize the system to better meet their needs.

In AniViz, users can use keyboard shortcuts to save or load instances. It could also have a menu-like interface that lets users save and load instances they create as projects. Furthermore, we could extend AniViz to let users import and export templates and instances. In this way, a group of scientists could collect and share templates or instances from different datasets.

For time-varying, multivariate data up to hundreds of gigabytes, our system on a PC is highly interactive except when browsing the time series, which requires intensive disk access. To improve the performance, we could use prediction, temporal-space encoding, and data prefetching. As shown in the videos at www.computer.org/portal/web/computingnow/cga/videos, specifying and assembling pieces of animation content (instances) normally takes several minutes. For data of more than hundreds of gigabytes, we could employ downsampling or multiresolution rendering to reduce the data to a suitable scale for interactive data exploration and animation content specification and

assembly. The original data, however, could still be used for high-quality output during the final animation production.

Finally, AniViz should be able to output more than just animation. On the basis of the key information attached to instances, the ideal output is multimedia, which integrates a wide variety of forms, such as image, video, text, and sound, for a more comprehensive presentation. Some studies on visual media provide inspiration in this direction (see the "Conceptual Models and Visual Media" sidebar).

Scientists commonly use animation to explain complex physical phenomena or chemical processes. The volume visualization examples we've presented show that AniViz offers scientists expressive power in making animation by using various options to create in-between frames. Because AniViz is tightly coupled with an interactive visualization system, animation can become a by-product of data exploration. Scientists can play back a sequence for fine-tuning or simply rearrange it without leaving the system. Our research shows the value of advanced animation support for scientists to make sophisticated animation sequences by themselves. We anticipate future visualization systems equipped with animation support comparable to AniViz.



Acknowledgments

This research was supported partly by the US National Science Foundation through grants CCF-0811422, CCF-0808896, OCI-0749227, OCI-0749217, CNS-0551727, and OCI-0325934 and by the US Department of Energy through the SciDAC (Scientific Discovery through Advanced Computing) program with agreements DE-FC02-06ER25777 and DE-FG02-08ER54956. We thank the reviewers for their helpful comments.

References

1. L. Moltedo and S. Morigi, "Anima: An Interactive Tool for Scientific Data Animation," *Computer Graphics Forum*, vol. 12, no. 5, 1993, pp. 277–288.
2. O. Arikan, D.A. Forsyth, and J.F. O'Brien, "Motion Synthesis from Annotations," *ACM Trans. Graphics*, vol. 22, no. 3, 2003, pp. 402–408.
3. "Scientific Visualization Studio," NASA, 2010; <http://svs.gsfc.nasa.gov>.
4. "Featured Visualizations," Univ. Corp. for Atmospheric Research, 2010; www.vets.ucar.edu/vg.
5. "MPEG Animations," Oak Ridge Nat'l Lab, 2010; www.csm.ornl.gov/viz/mpegs.html.
6. "ASC Visualization Project: Image and Movie Gallery," Lawrence Livermore Nat'l Lab, 2010; <https://computing.llnl.gov/vis/images.shtml>.
7. "Featured Animations," EnSight, 2010; www.ensight.com/featured-images/animations.html.
8. J. Clyne et al., "Interactive Desktop Analysis of High Resolution Simulations: Application to Turbulent Plume Dynamics and Current Sheet Formation," *New J. Physics*, vol. 9, no. 301, 2007; www.iop.org/EJ/abstract/1367-2630/9/8/301.
9. L. Perkins, "Tropical Storm Edouard," NASA, 2010; <http://svs.gsfc.nasa.gov/vis/a000000/a003500/a003536/index.html>.
10. H. Akiba and K.-L. Ma, "A Tri-space Visualization Interface for Analyzing Time-Varying Multivariate Volume Data," *Proc. Joint Eurographics/IEEE VGTC Symp. Visualization*, Eurographics, 2007, pp. 115–122.
11. A.S. Glassner, *Principles of Digital Image Synthesis*, vol. 1, Morgan Kaufmann, 1995.

Hiroshi Akiba is a component design engineer at Intel. His research interests include visualization, user interface design, and GPU computing. Akiba has a PhD in computer science from the University of California, Davis. Contact him at hiroshi.akiba@intel.com.

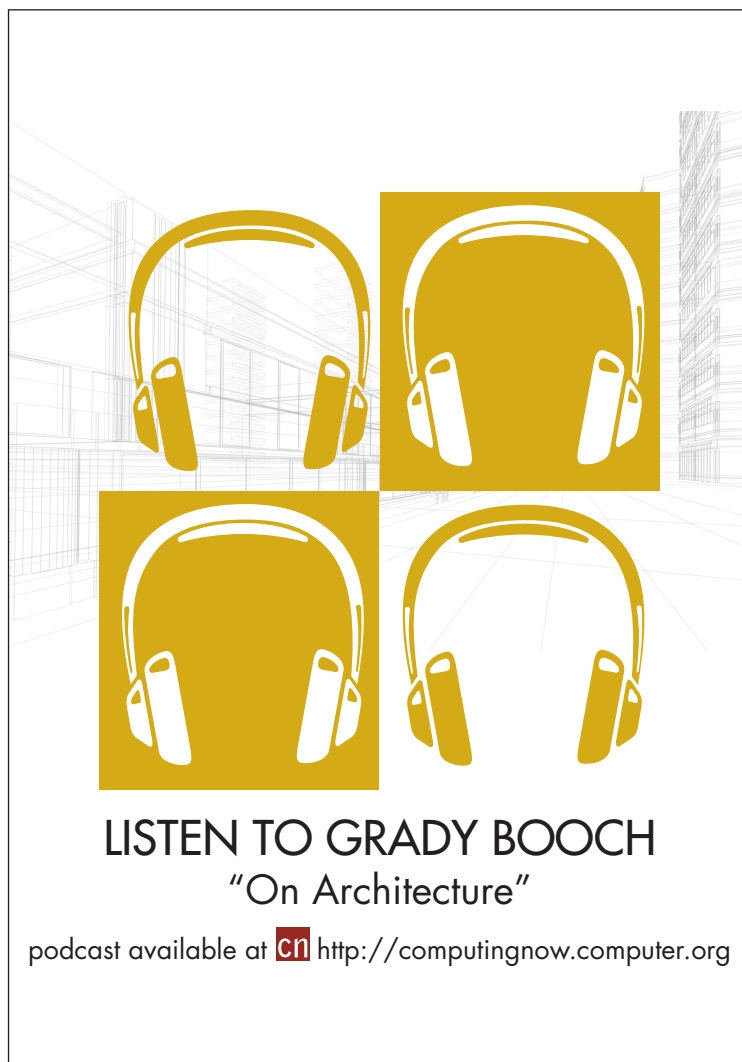
Chaoli Wang is an assistant professor of computer science at Michigan Technological University. He previously was a postdoctoral researcher at the University of California, Davis. His research focuses on large-scale data analysis and visu-

alization, high-performance computing, and user interfaces and interaction. Wang has a PhD in computer and information science from Ohio State University. He's a member of the IEEE. Contact him at chaoliw@mtu.edu.

Kwan-Liu Ma is a professor of computer science at the University of California, Davis and directs the US Department of Energy's SciDAC (Scientific Discovery through Advanced Computing) Institute for Ultrascale Visualization. His research involves visualization, high-performance computing, and user interface design. Ma has a PhD in computer science from the University of Utah. He serves on the editorial board of IEEE Transactions in Visualization and Computer Graphics and IEEE Computer Graphics and Applications. He's a senior member of IEEE and a member of the ACM. Contact him at ma@cs.ucdavis.edu.




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

The advertisement features a background image of a city street with buildings. Overlaid on this are four yellow squares arranged in a 2x2 grid. Each square contains a white silhouette of a pair of headphones. Below the grid, the text 'LISTEN TO GRADY BOOCH' is written in a large, bold, sans-serif font. Underneath that, the subtitle '“On Architecture”' is written in a slightly smaller, italicized font. At the bottom, it says 'podcast available at' followed by the 'cn' logo and the URL 'http://computingnow.computer.org'.

LISTEN TO GRADY BOOCH

“On Architecture”

podcast available at  <http://computingnow.computer.org>