Using image processing and computer animation to analyze spatial and temporal dynamics in a pattern-forming system

K. A. Robbins[a)]
Division of Computer Science, University of Texas at San Antonio, San Antonio, Texas 78249-0667

M. Gorman[b)]
Department of Physics, University of Houston, Houston, Texas 77204-5504

(Received 17 January 1997; accepted 11 September 1997)

The availability of inexpensive computer equipment for video acquisition and storage can improve the way data from systems with complex spatiotemporal dynamics are acquired, analyzed, and disseminated. Cellular flames exhibit ordered patterns that bifurcate to both periodic and chaotic states. While the general characteristics of the motion can be obtained from direct visual observation, the precise specification of the dynamics requires a more sophisticated quantitative analysis. The application of computer image processing and animation techniques can enhance the analysis of such dynamics. The article describes the entire process from experiment to dissemination and discusses the obstacles encountered at each step. © *1997 American Institute of Physics.* [S0894-1866(97)01906-8]

INTRODUCTION

The availability of inexpensive computer video-acquisition and storage equipment for desktop workstations has important implications for visualization and analysis of scientific data. Technology previously found only in supercomputer centers is now available on PCs. The accessibility of the World Wide Web makes it possible to communicate results using video. These technological developments present new opportunities for using image-processing techniques to display and analyze experimental data in video format.

The analysis of spatial and temporal dynamics in pattern-forming systems is an area that can benefit from this progress in desktop video technology. Spatiotemporal dynamics appears in many important physical applications in fluid dynamics, chemistry, nonlinear optics, condensed matter, and biology. The term spatiotemporal generally applies to systems that undergo loss of stability from a steady, spatially invariant state to states with complicated spatial structure as a control parameter is varied. These spatial structures in turn destabilize and evolve into other spatial states that may be steady or vary in time. The article by Cross and Hohenberg[1] gives an excellent overview of pattern formation in systems far from equilibrium.

Premixed flames have two distinctive characteristics that make them interesting for studies of pattern formation. First, standard optical and video techniques can be used to measure both the spatial and temporal characteristics of the dynamics because the chemiluminescence emitted at a point in the flame is directly related to the local temperature. Second, the cellular flame front undergoes bifurcations from an ordered pattern of concentric rings of cells to many (≈ 50) different dynamical states, each with its own spatial and temporal motion, providing many different examples of dynamics.

In this article we explain the issues of image processing and computer animation that arise in the analysis and display of this dynamics. We also describe the tools necessary to accomplish the analysis using a desktop video workstation. The starting point of the process is an experiment that is instrumented with video imaging equipment to acquire digital images either directly or through conversion from analog to digital format. Image processing is then used to isolate and quantify features, and the resulting data set is animated to produce simplified visual models of the behavior.

Three important differences make the implementation of computer visualization and animation for data from experiments in spatiotemporal dynamics more challenging than that for data from computation.

(1) Data from numerical simulations are usually generated in digital form and are relatively free from noise and external perturbations. The acquisition of experimental data typically involves a conversion from analog to digital introducing external noise that must be distinguished from the internal chaotic dynamics of the system being studied.
(2) The time step used in numerical simulations must be small relative to the time scale of the phenomenon, giving the researcher flexibility in selecting the time scale on which to visualize. Imaging of experimental data, by contrast, can be limited by the time scale of the imaging equipment (1/30 of a second for standard video cameras). Digital cameras with higher temporal resolution are available, but these systems are expensive and require a trade-off between temporal and spa-

[a)]E-mail: krobbins@runner.utsa.edu
[b)]E-mail: gorman@uh.edu

tial resolution because of the necessity of transferring data to a storage device. Complex spatiotemporal dynamics also requires a high spatial resolution to capture features of motion that may extend over multiple length scales.

(3) Numerical simulations often have complete state information available at each time step, allowing the superposition of multiple visualizations to extract computed quantities. Invariants, such as conservation of vorticity, can be calculated to assist in the detection of features and in modeling the phenomenon. Experimental visualization is usually based on partial state information. In some experiments it is possible to correlate measurements from multiple sources, but the availability of state information is usually relatively limited when compared with numerical simulations.

In this article we explore the use of visualization and animation of experimental data from an experiment in combustion. In Sec. I the experiment is discussed, and in Sec. II data-acquisition issues are addressed. In Sec. III we describe the image-processing environment that we selected for developing our tracking application and explain why system development in a standard environment is important. The actual details of the tracking algorithms are presented in Sec. IV. In Sec. V we explain how visualization and animation can enhance the insight gained from quantitative measurements, and in Sec. VI we discuss the dissemination of results and how future developments in technology will change the way data are modeled and disseminated.

I. THE EXPERIMENT

A premixed flame front is stabilized on a circular porous plug burner mounted in a combustion chamber.[2] The steady, flat flame front appears as a 0.5-mm-thick, uniform circular disk of light that sits 5 mm above the burner. As the control parameters are varied, either pulsating flames or cellular flames are observed. The dynamic behavior of the front is controlled by varying any of four experimental parameters—the ratio of fuel to oxidizer (the equivalence ratio), the total flow rate of the fuel–oxidizer mixture, the pressure in the chamber, and the type of fuel—with a resolution of 0.1% of full range. In this article the results on cellular flames using either isobutane–air or propane–air mixtures at a chamber pressure of 1/2 atm are emphasized.

At low pressure ($\approx 1/2$ atm) and moderate numbers of cells ($\approx 8-20$) the brighter, hotter cells form ordered patterns of concentric rings. As the flow rate and/or the equivalence ratio are varied, the ordered states undergo bifurcations to other ordered states with slightly different numbers of cells or to dynamic states in which cells or rings of cells move in a distinctive manner. Although the general characteristics of the motions can be observed directly, other techniques must be used to obtain quantitative descriptions of the behavior. Each state presents particular challenges. We now describe representative examples of the observed states and explain the pertinent questions of physics in each case.

In Fig. 1 four sequential frames of videotape are presented to depict the motion of representative states. In Fig. 1(a) an ordered state with 12 outer cells and four inner cells is shown. The ordered states are not steady; they are chaotic. The cells subtly change their size and shape while maintaining the overall pattern.[3]

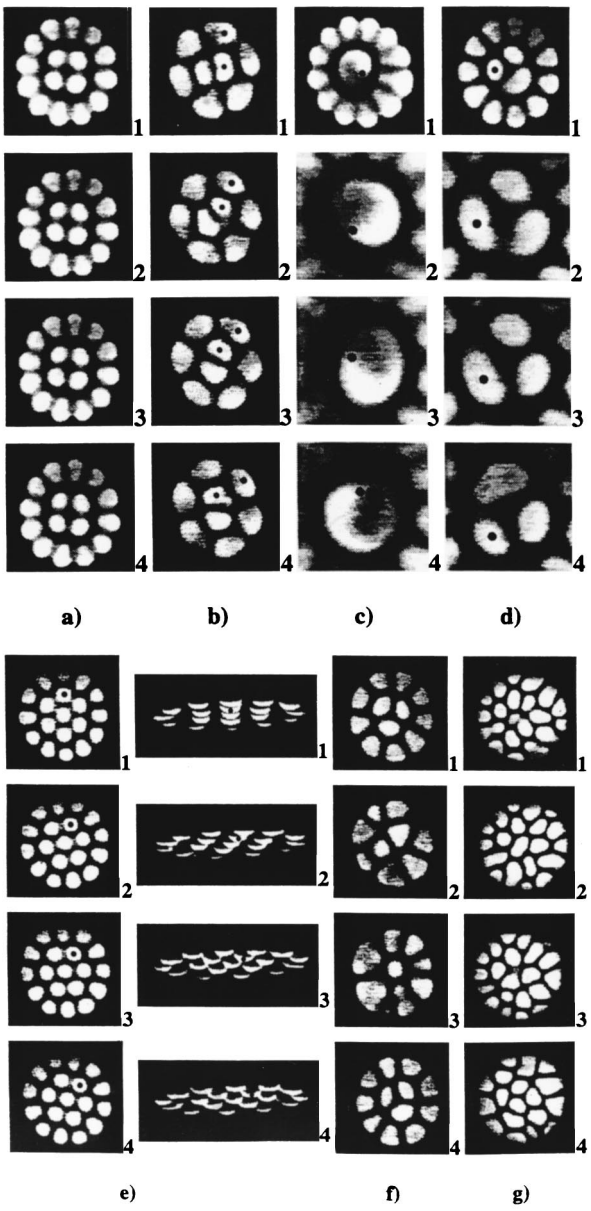

Figure 1. Four consecutive frames of video for (a) an ordered state; (b) a rotating state; (c) a rotating spiral state (in closeup); (d) a hopping state (in closeup); (e) a ratcheting state (in top and side views); (f) an intermittently ordered state; and (g) a disordered state.

The dynamic states typically bifurcate from ordered states with approximately the same number of cells. Two examples of rotating states are shown in Figs. 1(b) and 1(c). In Fig. 1(b) two concentric rings of cells counterrotate (2 Hz). A single inner cell assumes a spiral-like shape and uniformly rotates (3 Hz) as shown in Fig. 1(c). The asymmetry of the uniformly rotating cells is a signature of parity breaking.[4] In contrast, a steady pattern, such as the one

Table I. Comparison of data storage requirements for different video frame formats and image sizes.

Format	Image dimensions (pixels)	Image size (kbytes)	Storage 30 min (Gbytes)	Required transfer rate (Mbytes/s)
RASTER(24-bit)	640×480	920	49	27
RASTER(24-bit)	320×480	460	25	14
RASTER(8-bit)	640×480	307	16	9.2
RASTER(8-bit)	320×480	154	8	4.6
RASTER(8-bit)	320×240	75	4	2.3
RASTER (compressed)	320×240	40	2.2	1.2
JPEG	640×480	17	0.9	0.5

shown in Fig. 1(e), has a central cell that appears as a symmetric circle.

Figure 1(d) shows a closeup of a hopping state with 11 cells in the outer ring and three cells in the inner ring. Hopping cells move quickly from one stationary position to another in a way that appears disengaged from the motion of their neighbors. One hopping cell is marked with a dot to clarify changes in angular position. The hopping motion (6 Hz) proceeds sequentially to other cells in the ring. The numerical simulations[5] of the hopping states predict that the magnitude of all the angular displacements should be the same, but the experimental measurements show a substantial variation in this quantity. The presence of the fixed outer ring of cells may create a modulated structure that restricts the possible values.

A top and side view of a ratcheting state[6] is shown in Fig. 1(e). In this state the entire pattern of 12 cells around six cells rotates nonuniformly about a single central cell, speeding up and slowing down in a characteristic manner. The average angular velocity of the cells in ratcheting states is only 1 deg/s (<0.1 Hz). Accurate measurements of the angular position and angular velocity of each cell are crucial in describing the motion. In most of the 15 examples of ratcheting states found to date, computer animations are necessary to untangle the relative motion of the rings and to clarify the complicated nature of the nonuniform dynamics.

The side view of Fig. 1(e) shows the three-dimensional structure of cellular flames. The darker, cooler folds and cusps protrude upward, away from the burner and form the boundaries between the brighter, hotter cells. The dynamic states can be described in two equivalent ways: as the motion of concentric rings of cells or as the motion of ordered arrays of cusps and folds.

In Fig. 1(f) an ordered pattern of nine cells around three inner cells disappears in frame 2 and reappears at a later time in frame 4. These intermittently ordered states are characterized by the times the system spends in the various ordered states. Long time series of reasonable spatial resolution are necessary to measure the behavior of an intermittently ordered state. The resulting data sets may be longer than 30 Gbytes.

Figure 1(g) shows the temporal evolution of a disordered state. The cells move in an irregular motion, the rings of cells are no longer visible, and an ordered pattern is never observed. The onset of a disordered state demarks the maximum flow rate at which ordered states are observed for that equivalence ratio and pressure. The disordered states exhibit distinctive splitting and merging of cells.

The tracking of individual cells, the precise measurements of cell angular displacement and angular velocity, the classification of cell shape, the identification of ordered patterns, the detection of symmetries in individual cells and in the patterns of concentric rings, the visualization and quantification of the motion of cusps and folds that separate the cells, and the explication of the interaction of cells during complex motions each requires specific software and places unique demands on the computer-based data-acquisition system.

II. DATA ACQUISITION

In the current mode of operation, the top view of the flame dynamics is recorded on S-VHS videotape using a charge-coupled-device (CCD) camera. A time code is inserted directly into the video as an easy visual check to insure that each frame is digitized correctly. As a rule of thumb, S-VHS has an image resolution of 640×480 pixels, while VHS has an image resolution of 320×480 pixels.

Careful camera work is essential to insure a clearly distinguishable separation between the cells for successful cell tracking. For an analysis of the motion of the cells, the camera is focused to emphasize the contrast between the cells and their separating regions. This separation is especially important for executing long tracking runs that involve thousands of frames, because our tracking software requires operator intervention when the individual cells are not distinguishable. The three-dimensional structure of the front presents some challenges in determining the appropriate focal point for the camera. The darker cusps and folds protrude about 5 mm above the position of the uniform flame front. For an analysis of their motion the camera must be focused at their midpoint and the depth of field must be reduced to their height.

The proper characterization of a state with chaotic spatiotemporal dynamics involves the analysis of long runs at high spatial resolution. Grabbing individual frames in real time is not difficult, but grabbing a video clip of significant length is a challenge for existing off-the-shelf computer systems. Table I presents the data requirements in terms of several common image formats. Raster format is a raw image format consisting of a header followed by an array of pixels. In the 24-bit raster format, each pixel is represented by 3 bytes—red, green, and blue (RGB) values. In 8-bit raster format each pixel consists of a single byte representing an index into a color table. The image header contains

Table II. Sustained transfer rates for secondary storage devices.

Device	Sustained transfer Rate (Mbytes/s)
IDE disk drive	0.5
SCSI disk drive	0.5
Fast SCSI disk drive	2
RAID disk array	20

a color map, which is a table of RGB values indexed by the pixel values. Such a representation can accommodate up to 255 colors. Some systems represent images using an 8-bit compressed color format. This representation is slightly smaller than 8-bit raster. JPEG[7] is a single-frame compressed representation in which the image is divided into 8×8 blocks of pixels. Each block is transformed using a direct cosine transform, and only the most significant coefficients are retained. JPEG is a lossy compression scheme in that once coefficients are dropped they cannot be recovered when the image is uncompressed. The storage requirements for JPEG shown in Table I are based on standard quality parameters for lossy JPEG compression.

The fourth column of Table I shows how much storage is required for a 30-min clip. The transfer rates shown in the fifth column of Table I are based on digitization rates of 30 frames/s. The data requirements of Table I show that large clips are too big to be contained in main memory. Therefore, a video system must have fast, large capacity secondary storage to hold image data.

Table II presents transfer rates supported by various types of disk drives. Full-resolution 24-bit raster images require more bandwidth than can be sustained even with disk storage arrays. Although 9-Gbyte SCSI drives are inexpensive and widely available, operating systems for workstations and PCs are only starting to support disk partitions larger than 2 Gbytes. On the other hand, sequences of JPEG images fall within the transfer rate and storage capacity of ordinary SCSI disk drives.

Tables I and II show that with current desktop technology, digitization at acceptable resolution requires on-the-fly compression. A CODEC is the compression–decompression scheme used by such a system. Typical software CODECs include Cinepak, Microsoft Video 1, QuickTime, Indeo, and MPEG. Software CODECs do not require any special hardware, but they cannot compress in real time. Hardware CODECs perform on-the-fly compression, but these systems use compression schemes that are vendor-specific. The output generated by these boards must be played back on a system equipped with the same type of hardware.

For analysis, the output of the hardware CODEC must be converted frame-by-frame to a standard image format. We used a Parallax XVideo board in a Sun Sparcstation 10 as our hardware CODEC. The board digitizes color images of 640×480 pixels at 30 frames/s and stores the data in JPEG format in a single large movie file for subsequent processing as shown in Fig. 2. The digitization takes place in real time. In the second phase of processing, individual frames are extracted from the movie file, converted to raster format, and stored as separate consecutively number files in compressed format. Images are clipped and masked during this phase to minimize the storage requirement of the image files. Although the Parallax system came with a library of routines for manipulating raw movies, it was not designed to extract individual frames, so we had to write additional software to accomplish this task. Using the procedure described above, we can process 30-min video clips without difficulty. Longer runs are analyzed in pieces or at a lower sampling rate.

In addition to the Parallax system, we use a Bravado 1000 board running under Windows 95 in a Pentium Pro workstation. The board's CODEC is also based on JPEG. The resulting clips must be converted to AVI format and can then be edited using Adobe Premiere. This system is convenient for assembling clips and producing movies, but it cannot digitize at the resolution required without dropping frames.

The continual evolution of available hardware and software should make data acquisition easier in the future. However, this type of application tends to push the limits of system capacity, so it is important to use approaches amenable to handling large numbers of frames and to extracting models that simultaneously enhance understanding and reduce the size of the retained data sets.[8]

III. KHOROS 2 IMAGE PROCESSING DEVELOPMENT SYSTEM

Once the images have been converted and preprocessed, the cells are tracked using image-processing techniques. Since this work was viewed as a long-term project involving many types of analyses, we developed our software under an image-processing development system rather than writing standalone programs to do the image processing.

One argument for using such an environment is that the system provides a method for generating sophisticated

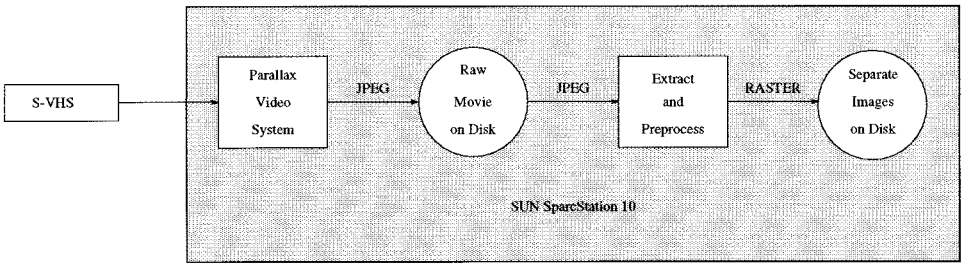

Figure 2. The video acquisition system and preprocessing setup. A raw movie file that can be acquired in real time is then separated into individual image files and preprocessed.

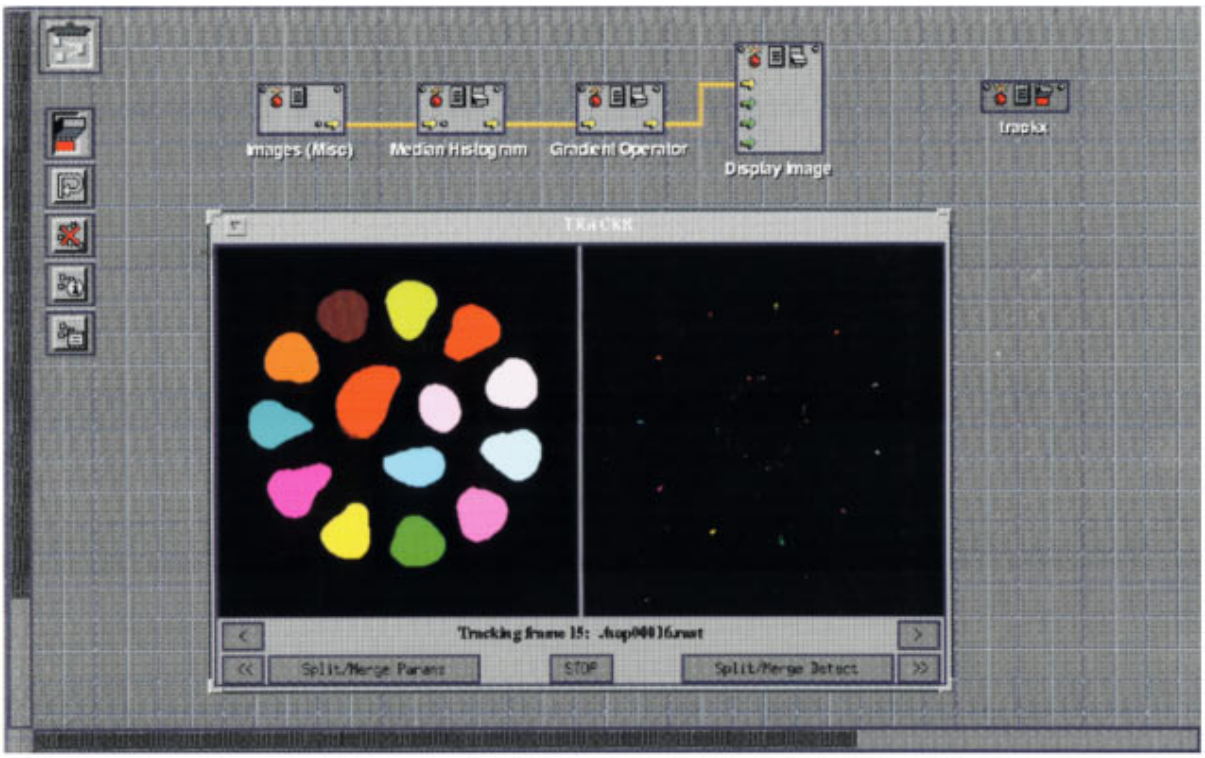

Figure 3. The Khoros visual programming environment supports development of user programs in a systematic manner. The double window at the center shows our trackx *program.*

graphical user interfaces (GUIs). Appropriate user interfaces are important, since the users of the programs are not the programmers who developed them. Another argument for this approach is that a development system usually specifies an application program interface (API) that allows different programs written in the system to work together in a reasonable way. The programs may be used for different projects over several years, and a development system promotes reusable code over the lifetime of the project.

The downside of such an environment is the considerable system administration needed to set up and to maintain the system. The development environment requires extra disk space, and programs run more slowly because of system overhead. Alternative environments such as MatLab or a commercial image-processing system were considered, but it was our judgment that having complete system source code and considerable program development support were essential for the project. We also wanted to avoid having to maintain a hodgepodge of programs that did not work together.

We selected the Khoros[9] image-processing environment originally developed at the University of New Mexico and now available over the Internet under a no-cost developers license for academic institutions. Khoros provides a common development environment so that individual programs can exchange data and work together in a reasonable way. Among its features are:

(i) support of common formats (raster, ppm, viff) and object-oriented representation of images;
(ii) Support for standard image-processing functions such as filtering, transforms and morphology;
(iii) additional toolboxes that can be easily added;
(iv) automatic generation of GUIs for user-written toolboxes;
(v) support of network transports for distributed processing.

Khoros programs run either as standalone programs with command-line arguments or in graphical mode from a visual programming language system called cantata. The large window in Fig. 3 is called the *user workspace* of cantata. The workspace can be scrolled, saved, and configured to the user's tastes. The individual functions or programs are organized into *toolboxes*. Several toolboxes come with the source distribution, and a large number of user-written toolboxes are freely available on the Internet.

Each program is represented in the cantata workspace by an icon called a *glyph*. A chain of four glyphs is shown in the upper left corner of the workspace in Fig. 3. The leftmost glyph selects a source image from the predefined set of images provided in the Khoros distribution. The second glyph represents a median filter, and the third glyph performs a gradient operation for edge detection prior to the display performed by the fourth glyph in the chain. A dot on an input (output) arrow indicates that the input (output) is available. The chain of glyphs represents successive transformations of the input image. By clicking the on switches in succession, the user initiates the transformations. The cantata environment also provides services for executing loops containing glyphs and for conditional execution.

The trackx glyph shown in the upper right corner of the workspace in Fig. 3 represents the program we devel-

oped to track individual cells through successive flame images. The display windows generated by trackx are shown in the bottom center of the workspace. The left subwindow shows the current image, and the right subwindow shows a time lapse of the center positions of the cells. The individual cells are color-coded and their colors remain the same through successive frames. A visual jump in the color indicates that the tracking program has failed to maintain tracking. The trackx program is the basis for the project's image-processing analysis.

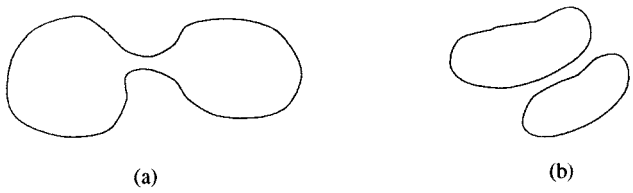

Figure 4. When a threshold is (a) too low an unwanted bridge forms, while if it is (b) too high a single cell may appear in two pieces.

IV. FEATURE-BASED TRACKING

Tracking is the key to quantitative analysis of cell motion,[10] and the analysis of the dynamic states requires the tracking of individual cells through thousands of frames. Tracking involves two steps: *segmentation* and *mapping*. Segmentation is the process of separating individual objects within an image. Mapping associates the objects in one image with the objects in another image.

The problem of segmenting objects within an image has received wide attention in the image-processing literature.[11] Some techniques rely on knowledge of the shape of the object or the fact that it is a rigid body such as a truck. Segmentation might use application-specific information such as texture or edge location to identify objects. The application here requires the segmentation of amorphous blobs.

Our trackx program performs segmentation using region growing combined with global dynamic thresholding to produce a *segmented image*. A segmented image is one in which pixels associated with a particular cell contain a value identifying the cell rather than an intensity value. The region growing is performed by constructing a segmented image as follows.

The segmented image, which is the same size as the original image, is initialized so that each pixel contains -1. The program examines each pixel in the original image starting in the top left corner and scanning left-to-right, top-to-bottom. If an image pixel is below a threshold, the corresponding pixel in the segmented image is set to zero. On the other hand, if the image pixel is above the threshold and the corresponding pixel in the segmented image is still -1, the pixel is part of a cell that has not yet been visited. In this case the program increments the cell count and sets the segmented image pixel value to the cell count. The program recursively visits all neighboring image pixels above the threshold in the original image and sets the corresponding pixels of the segmented image to the cell number. In this way, once the program encounters any pixel from a given cell, it visits all of the pixels of that cell before processing pixels from any other cells.

Once region growing has produced a segmented image, the number of distinct cells in each frame is determined. If there are too few cells in the image, it is likely that the threshold is too low so background pixels fall above the threshold allowing a bridge to form such as that shown in Fig. 4(a). If there are too many cells, it is likely that the threshold is too high, causing a single cell to appear to be split in two pieces as shown in Fig. 4(b). In either case the threshold is adjusted, and the process is repeated until the correct number of cells is found or a maximum number of iterations is reached.

Global iterative thresholding is effective when the cells are well separated and the overall illumination is relatively uniform. The cellular pattern has an intrinsic chaotic motion in which the cells change their curvature, resulting in changes in emitted light intensity. The threshold adjustment allows the tracking program to handle these frames in a natural way. Other methods of segmentation such as morphological operations or edge detection either are less efficient or less effective for these experiments. More sophisticated techniques are required when the emitted light intensity is nonuniform, the image illumination is uneven, or when the number of cells varies from frame to frame. These techniques are beyond the scope of this article.[12]

Segmentation produces an arbitrary labeling of the cells in an image, so the number assigned to a cell in one image may not correspond to the number of the same cell in another image. trackx associates corresponding cells in successive frames during the mapping phase. In slowly varying modes, the individual cells can be tracked by calculating the closest centroid of each cell in the next frame. In other modes, individual cells may move more than a cell radius. Cells may coalesce, bifurcate, or disappear altogether. Tracking in these more complicated cases uses a weighted measure of several parameters including distance from the centroid, overlap of cell bounding boxes, predicted position based on estimated velocity and acceleration, and positions of leading and trailing edges of the cells.[12]

Prior to the introduction of imaging-processing techniques, cell positions were measured manually by superimposing a polar grid on a video monitor and estimating the position of the leading edge of one cell. Image processing dramatically improved the accuracy and speed of the measurement process. The tracking program measures displacement of the cell centers for all cells in the front. Other programs compute displacements, velocities, correlations, and power spectra.

Figure 5 shows angular displacement versus time for all of the cells in a ratcheting mode. The plot of all angular displacements on the same graph conveys a global picture of the motion. The cells within each individual ring are highly correlated, and each ring undergoes a characteristic ratcheting motion. The inner ring rotates considerably faster than the outer ring, and the velocity maxima of the two rings appear to be correlated. In Sec. V different ways of visualizing the quantitative information that is extracted from tracking programs are discussed.

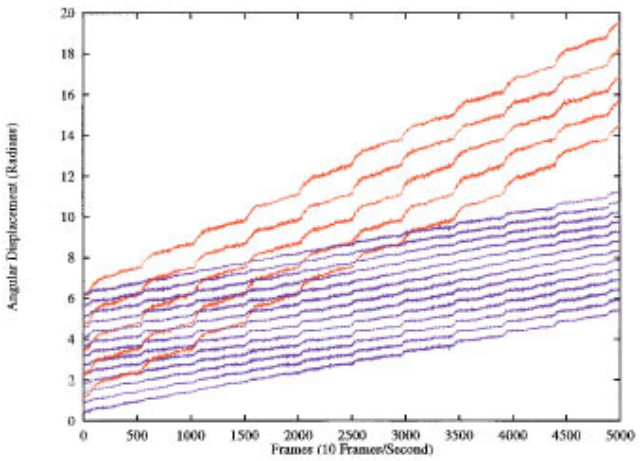

Figure 5. Plot of angular displacement vs time for a ratcheting mode in which two rings of cells move at different rates. The inner cells are plotted in red and the outer cells are plotted in blue.

V. VISUALIZATION AND ANIMATION

Spatiotemporal systems exhibit complicated dynamics, and visualization is an essential tool for understanding them. The rings of cells do not rotate as rigid bodies in the ratcheting states. Rather, there are internal motions of the cells in addition to the large-scale motion of the rings. Figure 6 uses a space-time diagram to illustrate small deformations in a ring. Each frame is represented by a striped vertical bar of constant height. The individual colored boxes in the bar represent scaled distances between the neighboring cells in the ring. If all the distances between cells are the same, the individual boxes within each stripe will be of exactly the same height. As time progresses to the right, the visual impression of waves of color shows the small azimuthal deformations that occur as the ring moves.

The bar graph of Fig. 6 represents time using the horizontal spatial axis of the plot. True time animations allow one to see particular relationships among geometric structures in motion. Figure 7 shows an animation of a ratcheting mode in which the inner ring moves and the outer ring is essentially stationary. The animation represents each cell as a node in a graph and connects the cells that are physically adjacent in the first frame. As the rings move, the graph deforms, showing the relative displacement of the rings.

Although software packages such as Mathematica and MatLab have some facilities for data animation, we found the visualization requirements for this problem to be quite specific. We developed most animations from scratch using Polka,[13] a C++ animation library with functions for creating and moving objects through time. Polka is event-driven, meaning that it increments its time to the point at which the next event occurs. The user can schedule multiple events to occur at any animation time. The following sample C++ code moves a circle object called cell from location (x_1, y_1) to (x_2, y_2) in two equal steps.

```
Loc From(x1, y1);
```

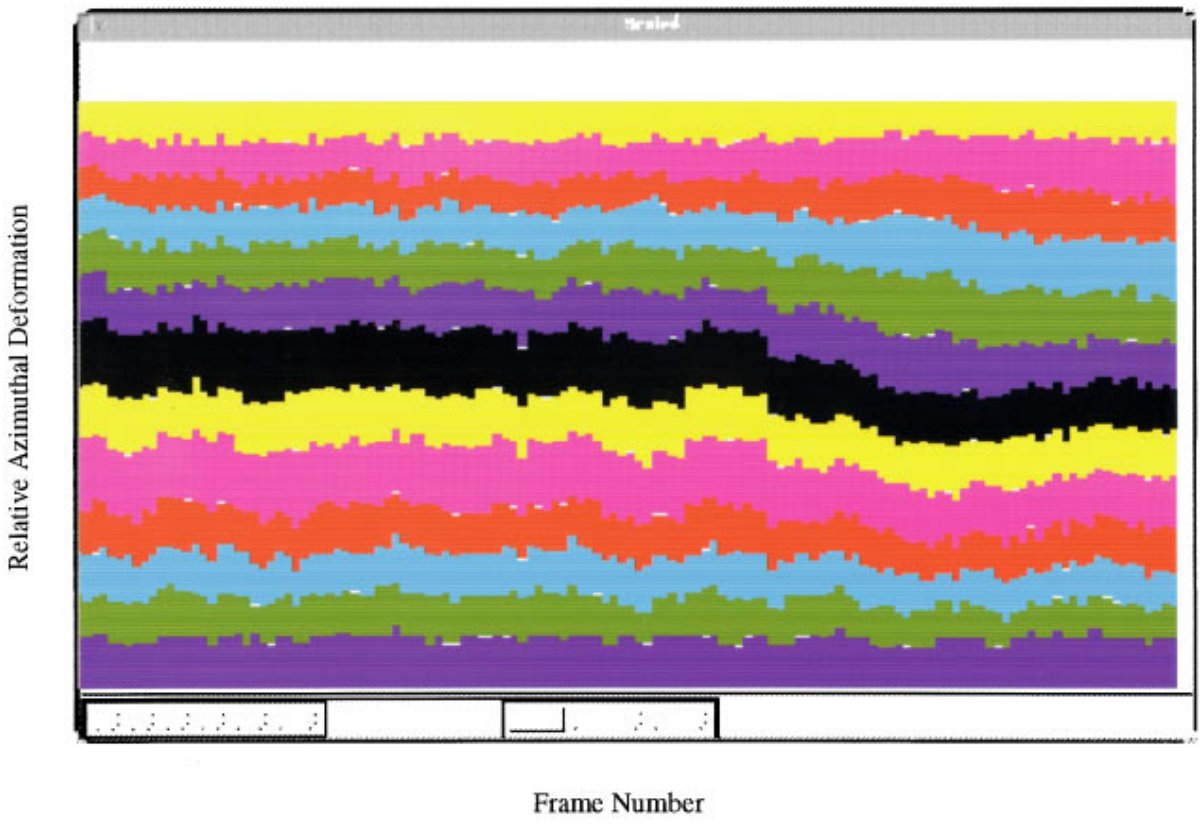

Figure 6. A space-time diagram showing azimuthal deformations of the ring.

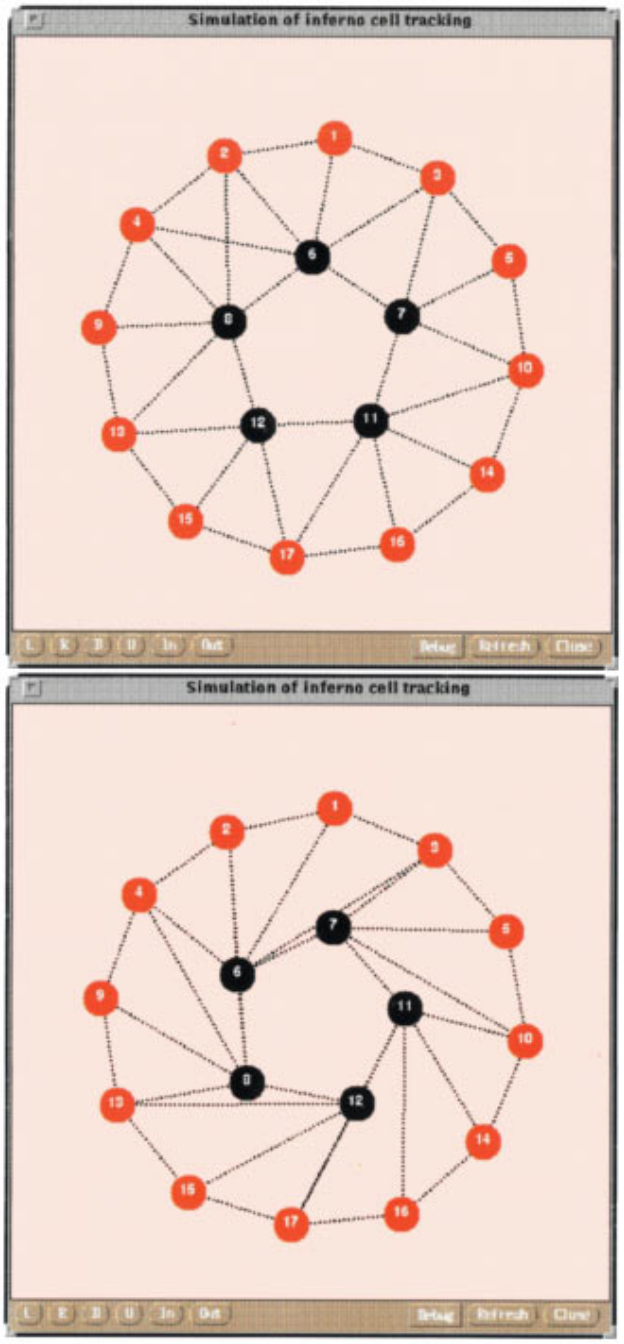

Figure 7. Two snapshots of an animation that models the flame front as a deforming graph. The top frame shows the initial position of the rings.

```
Loc To(x2, y2);
Action centermove("MOVE", &From, &To, 2);
len=cell→Program(time, &centermove);
time=Animate(time, len);
```

The first two statements set up location objects for the initial and final location of the circle. The Action statement creates a path along which to move the object. The Program statement schedules the cell object to move along the path specified by centermove. Polka schedules each step along the path at successive time steps starting with time. The actions do not actually take effect until the Animate is executed. The Animate call moves the internal animation time forward by len and updates the display according to the scheduled events.

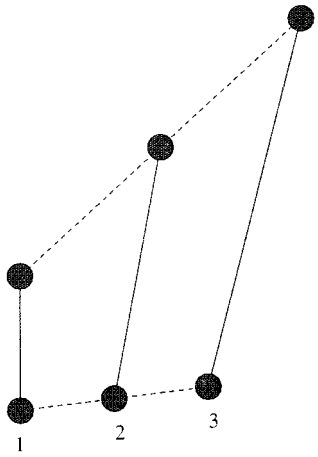

Figure 8. The steps involved in deforming the graph along a specified path in Polka.

Figure 8 shows the steps for deforming a graph consisting of two circles whose centers are connected by a line. The movements of the cell centers and of the line between them must be programmed individually. The motion of a circle object is specified by the position of its center along a path. A line object is specified by the position of an endpoint and a change in the x and y coordinates required to reach the other end. Motions for lines are specified in terms of paths for these variables.

In order to view a Polka animation, the user must be running the Polka animation system. Physicists at the University of Houston interacted with the animations by remotely logging into Unix machines at the University of Texas at San Antonio. In addition to the inconvenience of remote login, viewing of the animation was affected by the speed of the network connection and the ambient network load.

To make the animations more accessible, we have started to develop in Java, an object-oriented language designed by Sun Microsystems.[14] The programs are similar to those developed under Polka, but Java development has several advantages. Java is designed to be executed from a browser on the World Wide Web, so users who access that Web site can run the animations on their own machines once the data have been downloaded.

VI. DISCUSSION

The overall goals of the work described in this article are to study the spatial characteristics of chaotic dynamics and to develop techniques for analyzing systems that exhibit this behavior. A premixed flame stabilized on a circular burner is a prototypical system for studying spatiotemporal dynamics. It exhibits a myriad of ordered and dynamical states, and it has an intricate bifurcation structure of transitions among these states. Each problem encountered in the analysis of these dynamics requires the development of image processing, visualization, and animation tools. The

computerized quantitative frame-by-frame analysis of video data of considerable duration is a crucial capability. Accurate measurements of cell positions over thousands of frames also provide information about shape, size, and orientation of the cells. These measurements represent an improvement over manual techniques that only gave the position of the leading edge of a single cell.

The image processing produces huge data sets, and visualization is needed to make them accessible. One of the novel features of these experiments is the frame-by-frame analysis of long (2-h) data runs of highly irregular motion. The toolbox of techniques developed in these studies can then be used to analyze other physical systems with complex dynamics.

The stability boundary diagrams of the dynamic states are quite intricate, and many of the bifurcation sequences occur in small regions of parameter space. The dynamics at any particular set of parameters, total flow rate and equivalence ratio, is currently made by a judgement of the experimentalist based on visual identification of the motion. Most quantitative analysis occurs offline after the experiment is finished. Real-time identification and analysis would considerably enhance our ability to conduct the experiments by providing direct information on the dynamics while the experiment is in progress. These capabilities are also important for implementing methods of controlling the dynamics. We are currently integrating some real-time analysis capabilities into our acquisition system beyond standard point spectra and other traditional methods.

The visualizations and animations described in this article are best displayed in a video format. The increase in workstation computational power and the availability of affordable, sophisticated hardware and software will enhance the ability to project and access video anywhere on the planet. Several professional societies have initiated electronic versions of archival journals in which multimedia presentations can be embedded.[15] These developments will eventually change the standards for presentation of complex data and stimulate the development of more innovative approaches to spatiotemporal dynamics.

VII. SITES FOR ADDITIONAL INFORMATION

The World Wide Web site for this research is

http://vip.cs.utsa.edu/flames/overview.html

Internet sites of interest include:

Adobe Premiere: http://www.adobe.com
Bravado 1000: http://www.truevision.com
Khoros: http://www.koral.com/khoros/home.html
Java: http://www.java.com
Mathematica: http://www.wolfram.com
MatLab: http://www.mathworks.com
MPEG: http://www.bok.net/~ tristan/MPEG/MPEG-content.html
Parallax Graphics: http://www.parallax.com
Polka: ftp://par.cc.gatech.edu(130.207.119.254)

ACKNOWLEDGMENTS

This research was supported by Grant No. N00014-K-0613 from the Office of Naval Research. The authors would like to thank Balaji Natrajan for his work on a preliminary version of a trackx program and Jin Xu and Michael Villafana for assisting in the processing of the data. Devika Jarayaman programmed the adjacency animation and Joey Mukherjee programmed the bar graph animation. The authors would also like to acknowledge Steven Robbins for his help with the video hardware and software.

REFERENCES

1. M. C. Cross and P. C. Hohenberg, Rev. Mod. Phys. **65**, 851 (1993).
2. M. el-Hamdi, M. Gorman, and K. A. Robbins, Combust. Sci. Technol. **94**, 87 (1993).
3. M. Gorman, M. el-Hamdi, and K. A. Robbins, Combust. Sci. Technol. **98**, 37 (1994).
4. P. Coullet, R. E. Goldstein, and G. H. Gunaratne, Phys. Rev. Lett. **63**, 1954 (1989).
5. A. Bayliss, B. J. Matkowsky, and H. Riecke, in *Numerical Methods for PDE's with Critical Parameters*, edited by H. Kaper and M. Garbey (Kluwer Academic, Norwell, MA, 1992).
6. M. Gorman, M. el-Hamdi, and K. A. Robbins, Phys. Rev. Lett. **76**, 228 (1996).
7. G. K. Wallace, Commun. ACM **34**, 30 (1991).
8. D. Silver and N. Zabusky, J. Visual Commun. Image Represent. **4**, 46 (1993).
9. K. Konstatinides and J. Rasure, IEEE Trans. Image Process. **3**, 243 (1994).
10. R. Samtaney, D. Silver, N. Zabusky, and J. Cao, Computer **27**, 20 (1994).
11. A. Rosenfeld, Comput. Vision Image Understanding **62**, 90 (1995).
12. J. A. Withers, Master's thesis, University of Texas at San Antonio, 1996. (http://vip.cs.utsa.edu/flames/pubs/jawthesis/cellmap.html)
13. J. T. Stasko and E. Kraemer, J. Parallel Distr. Comput. **18**, 248 (1993).
14. K. Arnold and J. Gosling, *The Java Programming Language* (Addison-Wesley, Reading, MA, 1996).
15. The Institute of Physics has announced that two of its journals, *Combustion Theory and Modelling and Journal of Micromechanics and Microengineering*, will provide support for online multimedia in the electronic version of these journals on an experimental basis.