Frameworks for Visualization at the Extreme Scale

Kenneth I. Joy[1], Mark Miller[2], Hank Childs[2], E. Wes Bethel[3], John Clyne[4], George Ostrouchov[5], Sean Ahern[5]

1. Institute for Data Analysis and Visualization, University of California, One Shields Avenue, Davis, CA 95616-8562 USA. 2. Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551 USA, USA. 3. Computing Sciences Division, Lawrence Berkeley National Laboratory, The University of California, Berkeley, CA 94720-8139, USA. 4. National Center for Atmospheric Research, Boulder CO, USA. 5. National Center for Computational Sciences, Oak Ridge National Laboratory, One Bethel Valley Road, P.O. Box 2008 MS-6016, Oak Ridge, TN 37831, USA.
Website: www.vacet.org

Abstract. The challenges of visualization at the extreme scale involve issues of scale, complexity, temporal exploration and uncertainty. The Visualization and Analytics Center for Enabling Technologies (VACET) focuses on leveraging scientific visualization and analytics software technology as an enabling technology to increased scientific discovery and insight. In this paper, we introduce new uses of visualization frameworks through the introduction of *Equivalence Class Functions* (ECFs). These functions give a new class of derived quantities designed to greatly expand the ability of the end user to explore and visualize data. ECFs are defined over equivalence classes (*i.e.,* groupings) of elements from an original mesh, and produce summary values for the classes as output. ECFs can be used in the visualization process to directly analyze data, or can be used to synthesize new derived quantities on the original mesh. The design of ECFs enable a parallel implementation that allows the use of these techniques on massive data sets that require parallel processing.

1. Introduction

Recent scientific simulations on the world's fastest supercomputers have resulted in multi-field meshes consisting of almost 30 billion elements and producing hundreds of time steps. The challenge in visualizing results at these ``extreme'' scales is in dealing with the incredible volume and complexity of the resulting data. These data sets are so complex that visualizations performed using only the original output fields are not always the most informative. To overcome this problem, post-processing and visualization tools have been created over the years to interactively develop new *derived* fields [1] that allow a better understanding of the data.

In this paper, we illustrate the use of a new type of derived quantity called an *Equivalence Class Function (ECF)*. Simply stated, these functions group a collection of elements from a mesh M into a set of classes or groups, where the members of each class share some fundamental property. Typically, we use a rectilinear mesh M_g to define the groups. A summary value is computed for each cell in M_g by applying a *summarization* operator to the elements in these cells. The result is a mesh of classes, and function defined over these classes. These function values can be mapped back to the original mesh, creating a *derived* data set that can be visualized, or combined with other data sets to help analyze the data.

ECFs can be used in the visualization process in two fundamentally different ways:

- Using the reduced representation defined by the mesh M_g, and the summary operator defined over the mesh, an ECF can be treated as a reduced representation of the original data set and visualized directly -- using the ECF in an *analysis*-oriented manner. When used for analysis, ECFs can reduce massive data to something that is more manageable and meaningful.

- An ECF can be used to synthesize new derived quantities on the original mesh -- using it in a *synthesis*-oriented manner. Synthesized ECFs can be visualized directly, or can be used together with other derived meshes to produce new visualizations that better represent the features in the data.

ECFs have applications ranging from standard visualization techniques to data discovery and analysis to uncertainty analysis. They provide a general framework where we are able to characterize reduced-resolution representations of the data, along with a synthesis-oriented method to generate statistically-based derived data quantities. The novelty of this work is the framework, which combines these two needs, and allows integration into other data-flow-network-based derived data frameworks.

Our definition and implementation of ECFs accommodates a data parallel implementation, which makes them applicable to the needs of applications that treat large or massive data. With their flexibility and data-parallel implementation, ECFs become a powerful tool for data analysis and visualization of complex data sets at the extreme scale.

2. Related Work

An important concept in visualization is that of derived fields. A derived field is a field whose values are computed from one or more other fields. Systems that utilize derived fields allow the end user to compute an extended, perhaps more meaningful, set of field values from the fundamental solution values typically saved with a simulation.

Moran and Henze [1] present a system that allows an end user to write arbitrary expressions to define new fields, and then apply a variety of visualization techniques to the result. Their ``Demand Driven Visualizer'' used a lazy evaluation strategy to speed the computation of these derived fields, and allowed an interactive specification of the functions generating the derived fields. McCormick et al. [2] presented a hardware-accelerated system providing similar capabilities and exploiting current graphics hardware for portions of the computational tasks that would otherwise be executed on the CPU. Childs et al. [3] describe VisIt, a turnkey system for large-scale visualization, which also has a subsystem dedicated to generating derived data sets. Their contract-based system enables the visualization of massive data sets in a data parallel environment -- greatly extending the size of the data sets that can be addressed.

ECFs are related to bin-smoothed scatter plots [4,5], which are used in the information visualization community. In addition, several researchers have utilized similar techniques to address flow visualization problems using linked views or brushing techniques [6,7,5]. When the ECF summarization operator is *average*, an ECF will result in a bin-smoothed scatter plot also known as a *regressogram.*

ECFs are a general concept that allows statistical summarization using a wide-variety of summarization operators. Within this single framework, we are able generate ECFs for data reduction purposes, and also synthesize new derived data sets that can be combined with other data sets to better analyze a simulation. Whereas most statistical packages break down on large or massive data, ECFs allow a data parallel implementation that enables one to address data sets at the ``extreme'' scale. The extreme-scale data set parallel context is a new environment for analysis techniques.

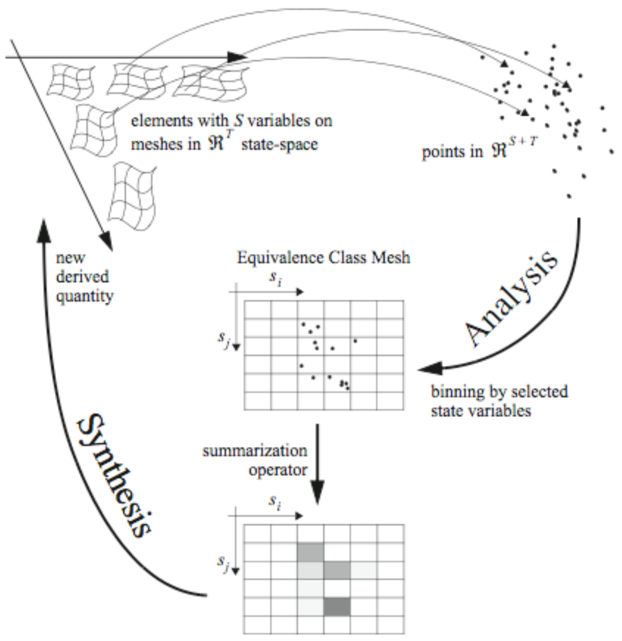
3. Equivalence Class Functions

An equivalence class function is constructed by applying basic analysis to group a collection of elements from a mesh M into a mesh of classes or groups M_g. A summary value is computed for each group of M_g by applying a *summarization* operator to all the elements of the group. The result can be viewed in two ways:

- as a function defined over mesh M_g, which can be used as a reduced representation of the original data set, or

- as a new derived data set over M, which is constructed by assigning the summary values to the corresponding points of the original mesh.

The diagram in Figure 1 illustrates these concepts.



Here, elements from input meshes (upper left) are mapped into a high-dimensional space (upper right) defined by the simulation's state variables. This cloud of elements is partitioned into classes (the analysis step), summary values are computed for each class (the summarization step), and the resulting mesh and field can be used to synthesize new derived quantities.

If we assume that our data sets are multi-valued, implying that the *dependent variables* in a simulation consist of a number of scalar values, vector values, and perhaps other quantities. The dependent variables define the *range* of a mapping from a *domain* defined by the *independent variables*. Typically, the independent variables are simply the spatial coordinates of the underlying mesh, or time. We refer to the union of the dependent and independent variables of a data set as the *state variables* of the data. Any derived quantities that are defined on the original mesh can be viewed as additional state variables. This allows for derived quantities, including those that are synthesized from an ECF, to be included in other ECFs.

The underlying mesh for an ECF, the *Equivalence Class Mesh* (ECM), partitions the element cloud into a uniform or rectilinear arrangement of groups (or bins). We begin its construction by selecting a subset of the indices of the state variables to address, and define the equivalence class mesh for the selected state variables as the Cartesian product of these sets of intervals. The ECM partitions a subspace of the element cloud into a uniform or rectilinear arrangement of groups (or bins). Each group holds a variable number of points. Some may be empty. The *value* assigned to a given point in the ECM involves applying a *summarization* operator to the set of points in each group and computing a single, summary value for each class. The possibilities for candidate summarization operators are almost limitless and vary significantly depending on the analyst's needs.

A common use case is to select one component (*e.g.*, one state variable) and then average this value over all members of each group. If the domain of the ECF is defined using the original mesh's independent variables, then this form of summarization yields a reduced resolution representation of the original mesh.

We can also compute summary values in a way that emphasizes some members in a class over others. For example, we can perform a volume-weighted average, or if we are performing an analysis over time, we can weight elements based on distance in time from an important event.

We can also use the rank of a group (e.g. the number of members) as a summarization operator. The resultant ECF has a number of interpretations relevant from a statistical standpoint. In the case of a one-dimensional domain, this form of summarization yields a histogram. In multiple dimensions, the resulting ECF is a multivariate histogram estimate of the joint probability density function of the selected state variables.

The summarization process need not result in a single scalar value for each class. If we are summarizing vector data, for example, we can compute a summary *vector* for each class. A more complex computation might be to fit a local statistical model to the data in a class and report the model parameter estimates as the vector for that class.

For synthesis, the summary value computed for a given ECM bin is mapped back onto the original mesh by assigning that value to every element in the associated class. This creates a new derived field on the mesh, which can be combined with other fields to produce visualizations that better illustrate the data.

4. Data Parallel ECFs

In a data-parallel model, it is generally assumed that the original data is decomposed into pieces that are stored in multiple files on disk [11]. In parallel, although the data-flow pipeline on each processor is identical, each pipeline operates on a different set of mesh pieces in streaming fashion.

We have designed the ECF system as a set of filters. An ECF filter accepts the original mesh data at its input and produces an ECF (mesh and field) at its output. The execution of the filter is governed by a set of user-specified control inputs.

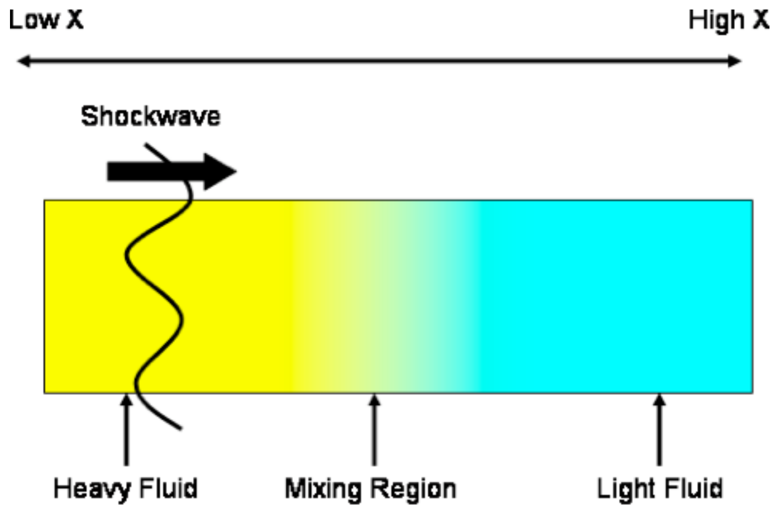
While the input mesh is decomposed across processors, the output is not. Instead, each processor maintains its own instantiation of the output ECM. Each processor computes its output ECM from the input mesh pieces that it owns. (Duplicating the output ECM on each processor is acceptable because the underlying ECM is rectilinear and typically much smaller than the original input mesh.)

After an initialization step, each processor enters a loop to iterate over all the input mesh pieces it has been assigned. On each iteration, a processor reads a mesh piece from a file on disk as well as the relevant state variables and then iterates over all elements updating the output ECF in the appropriate ECM bins. After all processors have processed their subset of the mesh pieces, each processor has a partial result. That is, each processor has a complete ECM populated with contributions from only the mesh pieces that it owns.

A parallel merge is required to compute the final result. In analysis-oriented use, the merged result can be created at the root processor only and the computation is complete. For synthesis, the summary value computed for a given ECM bin is mapped back onto the original mesh by assigning that value to every element in the associated equivalence class. This involves streaming the mesh pieces through the ECF filter a second time to find out which ECM bin each element of the original mesh lands in and then assigning the summary value for that ECM bin to the corresponding element.

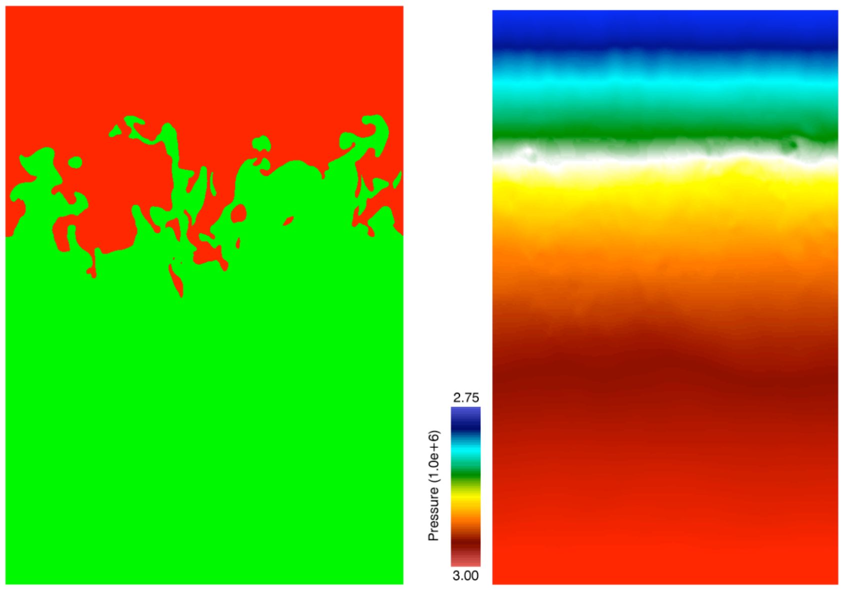
5. Results

ECFs have been implemented in VisIt [11,3] an end-user application for scientific visualization and data analysis. VisIt utilizes data flow networks and has an implementation that resembles and also leverages portions of VTK [12]. It has a distributed memory, data-parallel, data-flow execution model, which enables it to handle data at the extreme scale. It has an extensible derived function system in which we were able to implement our ECF framework.

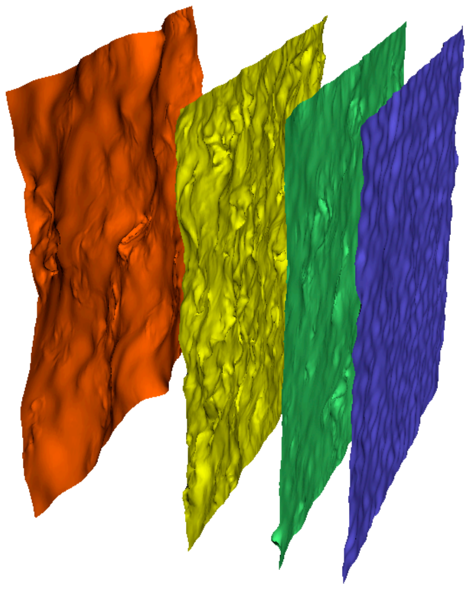


A Richtmeyer-Meshkov calculation simulates the mixing of heavy and light gases (fluids) as a shockwave passes through them. A typical orientation, one in which the shock wave propagates in the positive *x* direction passing first through the light gas to the left and then through the heavier gas to the right, is illustrated in the Figure above.

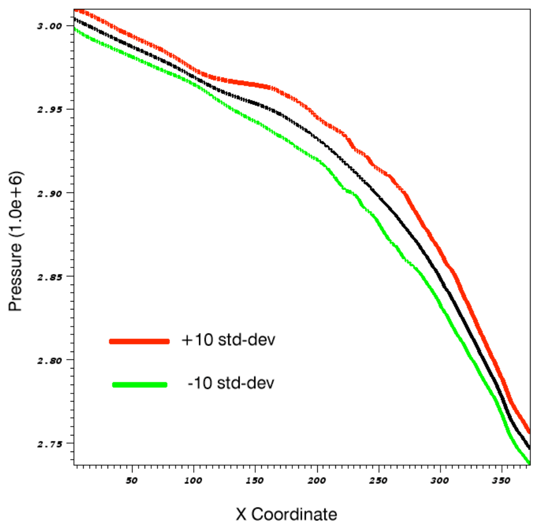
Typically, analysts study the *pressure* variable. The figures below demonstrate the application of standard techniques to visualize pressure from a 25 million element Richtmeyer-Meshkov calculation. The leftmost figure shows the interface between light and heavy gasses on the mid-*xy* slice, while the rightmost figure illustrates the pressure variable on the mid-*xy* slice.



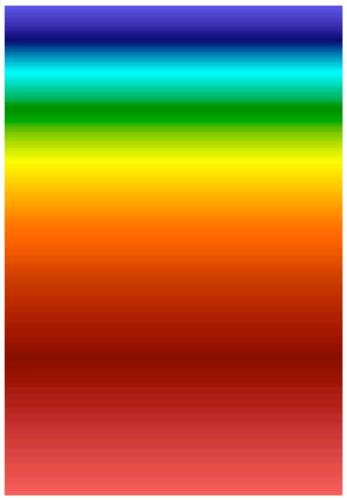
The figure below shows a set of isosurfaces of constant pressure. The isosurfaces tend to appear as parallel sheets with small deformations. Much of the turbulent mixing detail in the data is not evident from the images produced by these standard techniques.



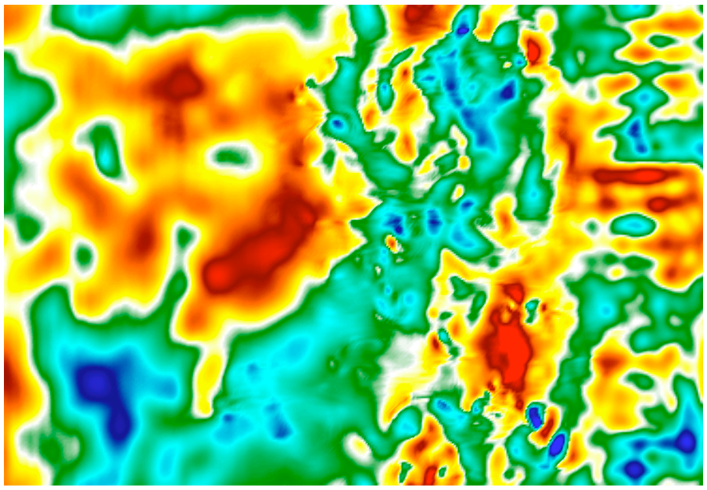
An important analysis technique for a three-dimensional Richtmeyer-Meshkov calculation is to compute the *transverse average*. The transverse average is the average pressure, over all *y* and *z*, as a function of *x*. It is straightforward to compute the transverse average using ECFs by selecting the *x*-coordinate field as the state variable over which to generate the ECM. The summarization operator is taken over pressure. The Figure below illustrates this one-dimensional ECF visualized directly



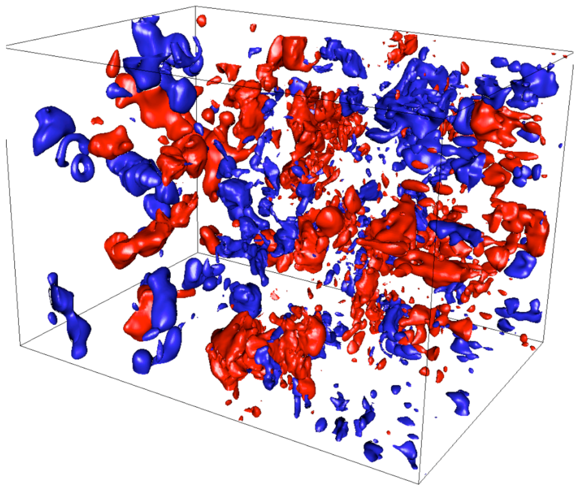
and the Figure below shows an *xy*-slice of its synthesis back onto the original mesh.



Given the transverse average and standard deviation ECFs and their syntheses, we can define a new derived quantity representing the number of standard deviations away from the transverse average for every element in the original mesh. Visualizing a slice of this derived data set, shown below, illustrates much more of the details of pressure variation, as we can identify those areas (in red) that are a significant deviation from the average.



We can also show the isosurfaces on the complete three dimensional data set (not only on a single slice) showing standard deviations from the transverse average pressure, as shown in the following figure. The red cells are those that show maximum deviation from the transverse average, while the blue cells show a lesser deviation.



Using these ECF-derived quantities gives much more insight to this complex data. Through the use of ECFs we can create new derived data sets and create new meaningful quantities by deriving new data sets from these ECF-generated quantities. By utilizing these methods, the scientist can better examine complex data.

6. Conclusion

In this paper, we have discussed equivalence class functions as a new framework that enables data analysis and visualization of complex data sets. We use equivalence classes meshes to partition the element cloud using basic statistical analysis, and apply summary operators to the bins of the mesh that generate values that can be mapped back to the original mesh. The framework allows the direct rendering of the equivalence class mesh -- a reduced representation of the data, or enables the synthesis of new derived data sets that can be used to better visualize the data. The novelty of this work is the framework, which combines these two needs, and allows integration into other data-flow-network-based derived data frameworks.

Our definition and implementation of ECFs also accommodates a data parallel implementation, which makes them applicable to the needs of applications that treat extreme-scale data. When integrated and used in concert with standard visualization techniques, ECFs offer a powerful way to manipulate, visualize and analyze scientific data. Formal definition of ECFs and their efficient data-parallel implementation for data sets at the extreme scale will extend the application of many statistical methods to data set sizes they previously could not reach.

References

[1] P. J. Moran and C. Henze, “Large field visualization with demanddriven calculation,” in Proceedings of IEEE Visualization, pp. 27–33, 1999.

[2] P. S. McCormick, J. Inman, J. P. Ahrens, C. Hansen, and G. Roth, “Scout: A hardware-accelerated system for quantitatively driven visualization and analysis,” in Proceedings of IEEE Visualization, pp. 171–178, IEEE Computer Society, 2004.

[3] H. Childs, E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. J. Whitlock, and N. Max, “A contract-based system for large data visualization,” in Proceedings of IEEE Visualization, pp. 190–198, 2005.

[4] M. O.Ward, “Xmdvtool: integrating multiple methods for visualizing multivariate data,” in Proceedings of IEEE Visualization, pp. 326–333, IEEE Computer Society Press, 1994.

[5] R. A. Becker and W. S. Cleveland, “Brushing scatterplots,” Technometrics, vol. 29, no. 2, pp. 127–142, 1987.

[6] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung, “Weave: A system for visually linking 3-d and statistical visualizations applied to cardiac simulation and measurement data,” in Proceedings of IEEE Visualization, pp. 489–492, IEEE Computer Society, 2000.

[7] R. M. Kirby, H. Marmanis, and D. H. Laidlaw, “Visualizing multivalued data from 2d incompressible flows using concepts from painting,” in Proceedings of IEEE Visualization, pp. 333–340, IEEE Computer Society Press, 1999.

[8] N.-A. Kumbah and E. J. Wegman, “Data compression by geometric quantization,” Computing Science and Statistics, vol. 27, pp. 284–288, 1996.

[9] J. M. Chambers, *Statistical Models*. CRC Press, Inc., 1991.

[10] D. Downing, V. Fedorov, W. Lawkins, M. Morris, and G. Ostrouchov, “Large data series: Modeling the usual to identify the unusual,” Computational Statistics and Data Analysis, vol. 32, pp. 245–258, 2000.

[11] H. Childs and M. Miller, “Beyond meat grinders: An analysis framework addressing the scale and complexity of large data sets,” in SpringSim High Performance Computing Symposium (HPC 2006), pp. 181–186, 2006.

[12] W. J. Schroeder, K. M. Martin, and W. E. Lorensen, “The design and implementation of an object-oriented toolkit for 3d graphics and visualization,” in Proceedings of IEEE Visualization, pp. 93–99, IEEE Computer Society Press, 1996.

[13] D. Scott, *Multivariate Density Estimation: Theory, Proactive and Visualization*. Wiley & Sons, 1992.

[14] Y. Matias, J. S. Vitter, and M. Wang, “Wavelet-based histograms for selectivity estimation,” SIGMOD Rec., vol. 27, no. 2, pp. 448–459, 1998.