Approximation of Time-varying Multi-resolution Data Using Error-based Temporal-spatial Reuse

Christof Nuber[a], Eric C. LaMar[b], Bernd Hamann[a] and Kenneth I. Joy[a]

[a]Center for Image Processing and Integrated Computing, Department of Computer Science, University of California, Davis, One Shields Avenue, Davis, CA 95616, U.S.A.
[b]Lawrence Livermore National Laboratory, Livermore, CA 94550, U.S.A.

ABSTRACT

We extend the notion of multi-resolution spatial data approximation of static datasets to spatio-temporal approximation of time-varying datasets. By including the temporal dimension, we allow a region of one time-step to approximate a congruent region at another time-step. Approximations of static datasets are generated by refining an approximation until a given error-bound is met. To approximate time-varying datasets we use data from another time-step when that data meets a given error-bound for the current time-step. Our technique exploits the fact that time-varying datasets typically do not change uniformly over time. By loading data from rapidly changing regions only, less data needs to be loaded to generate an approximation. Regions that hardly change are not loaded and are approximated by regions from another time-step. Typically, common techniques only permit binary classification between consecutive time-steps. Our technique allows a run-time error-criterion to be used between non-temporally consecutive time-steps. The errors between time-steps are calculated in a pre-processing step and stored in error-tables. These error-tables are used to calculate errors at run-time, thus no data needs to be accessed.

Keywords: Multi-resolution, Time-varying Datasets, Adaptive Rendering, Interactive Rendering, Error-based Rendering, Temporal Reuse

1. INTRODUCTION

Computer simulations of complex physical phenomena generate extremely high-resolution and more and more commonly time-varying datasets. Few of these datasets can be loaded into a computer's main memory. Sometimes even a single time-step is too large. Loading one time-step at a time can lead to unacceptable delays in applications where real-time system response is necessary. With a fixed budget assigned to loading and rendering data, mechanisms must be developed to meet these requirements. When compared to updating complete time-steps, multi-resolution representations of data in both time and space reduce the amount of data to be updated for each cycle.

Standard techniques used to explore time-dependent datasets have several disadvantages. Some use a binary classification, determining whether a node needs to be replaced (*i.e.*, whether it differs from the current time-step) or not, resulting in an error-based approximation where the error has been pre-defined during pre-processing. Techniques reusing already generated imagery instead of the underlying data replace only those image parts where the image needs to be updated. These approaches are usually not view-dependent. Common to all techniques is the notion of errors between consecutive time-steps only. Some of them are based on the assumption that there are only changes in small regions of the dataset or small changes between time-steps. Most existing techniques also use a pre-defined error during pre-processing.

Further author information: (Send correspondence to Christof Nuber)
Christof Nuber: E-mail: cnuber@ucdavis.edu, Telephone: 1 530 754 9470
Eric C. LaMar: E-mail: lamar1@llnl.gov, Address: Lawrence Livermore National Laboratory, 7000 East Ave., Livermore, CA 94550-9234, U.S.A.
Bernd Hamann: E-mail: hamann@cs.ucdavis.edu, Telephone: 1 530 754 9157
Kenneth I. Joy: E-mail: joy@cs.ucdavis.edu, Telephone: 1 530 752 1077

Our technique is more general. It considers data values instead of average values or images, which makes our approach view-independent. Instead of relying on a static binary classification to decide whether or not a region needs to be replaced, we provide an interactively user-defined error, replacing regions only when they exceed a user-defined error-threshold. With temporal-spatial reuse of subregions, we are not limited to consecutive time-steps. We allow a subregion from one time-step to approximate a congruent subregion in any other time-step, making no requirements with respect to spatial or temporal extent of changes in the dataset.

We avoid reloading of complete new time-steps by subdividing the dataset and reloading on a "replace/subdivide-where-necessary" basis, following a time-adaptive multi-resolution approach. In regions of the dataset where there is no change between time-steps, it is not necessary to replace that sub-region for a new time-step. The old region can be reused without error. Furthermore, we allow a sub-region from one time-step to approximate the same sub-region of a different time-step.

We use an N-ary tree decomposition of space and resolution for individual time-steps (*i.e.*, quadtrees for 2D data and octrees for 3D data). Leaf nodes contain original data and internal nodes contain approximations, with the root node containing the coarsest approximation. Each node is associated with a discrete (either original or approximated) sub-region of the dataset.

For each node we calculate in a pre-processing step its error-table, which will be used to determine the error between two arbitrary time-steps for that node. This approach allows us to determine the approximation error for any given node without accessing the original data during run-time.

Our technique complements (and can be used with) compression and differential encoding methods or other schemes used to reduce/compress data. The technique presented here is not restricted to 2D datasets, it is dimension-independent and applicable to n-dimensional time-varying datasets.

2. RELATED WORK

Finkelstein et al.[1] were among the first to develop methods for adaptive time-varying animation. They used a multi-resolution image technique and a quad-tree to decompose an image. Nodes represent flat regions (*i.e.*, they store the average color of the children) and may contain child pointers when higher-resolution information is available. The time dimension is encoded as a time-spanning binary tree, where one image quad-tree is associated with each node of the binary tree. The root node of the binary tree represents the averaged image from all time-steps (*i.e.*, a "motion-blurred" image). The leaf nodes of the binary tree store the imagery from a single time-step. There is no notion of run-time error, only the error associated with the initial construction of the time sequence is considered.

LaMar et al.[2] and Weiler et al.[3] described interactive multi-resolution volume visualization systems. A volume is composed into an octree hierarchy of approximations, with each level of the tree half the spatial resolution of the next level. The coarsest approximation is stored at the root node, approximations in the internal nodes, and the original data at the leaf nodes. A user-defined "importance" function, a suitable approximation error function, and rendering budgets guide the approximation.

LaMar et al.[4] introduced a fast method for computing error for multi-resolution volume rendering. The error is computed on the image volume produced by applying a color and opacity transfer function. Though the color space and data volume are large, for datasets with byte voxels there are only 256^2 unique pairs of error terms. The frequency of error terms in a node of the octree is recorded in a table, and it is also associated with that node. To evaluate the error associated with a node, only the table must be evaluated.

Ma et al.[5] used voxel-level quantization, octree encoding, and differential encoding for compression on voxel, spatial, and temporal dimensions, respectively. They claimed a compression of up to 90% for some datasets. Since their method uses voxel-level quantization, this is a lossy technique and is not appropriate for all visualizations. They used a technique called "temporal merging," where two or more temporally consecutive sub-regions are merged. The partial image generated from the first sub-domain of this series is cached and reused for rendering later time-steps. This approach has a limited notion of error with respect to time: A sub-region is reloaded when there is change in data values, and reused when there is no change.

Shen et al.[6] introduced the Time-Space Partitioning (TSP) method that decomposes space using an octree, and in each node of the octree, decomposes time using a binary tree. The TSP mechanism can represent, to a limited degree, error in both spatial and temporal senses. Non-leaf nodes store error terms, not approximations of data. "Bricks" of original data are stored only at the leaf nodes. Spatial error is used to determine when to approximate a sub-region of a volume with a single color, *i.e.*, a subregion is rendered either at original resolution or is approximated by a single color. Temporal error is used to determine whether imagery generated at one time-step can be used to approximate imagery of later time-steps. In both cases, error is estimated using the variance of spatial and temporal subregions. Low variance results in a constant color or reuse of an image; high variance results in rendering original data. Once an image is generated for a subregion, it is cached for possible reuse. Ellsworth et al.[7] extended the TSP algorithm to hardware texture-based volume rendering. They utilized alternate error metrics, *i.e.*, instead of examining data values they used the color values resulting from the transfer function.

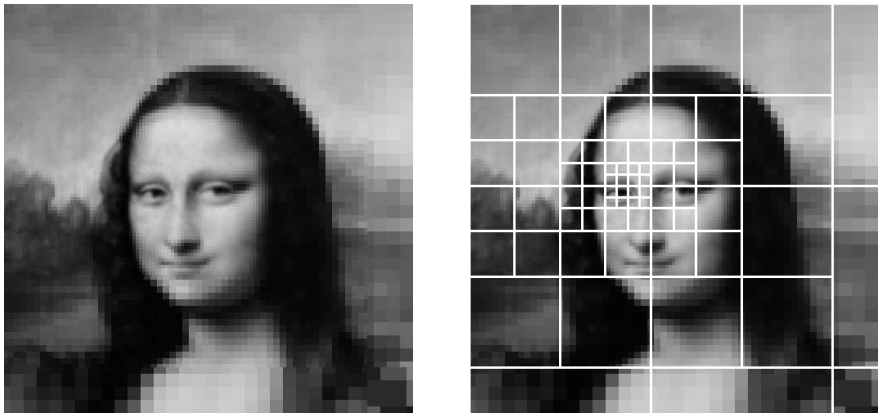
Sutton and Hansen[8,9] introduced T-BON, which is a time-based extension of the BONO (Branch-On-Need-Octree)[10] method for isosurface identification. The BONO method is an octree decomposition of space, where each node stores the extrema (minimum and maximum) over the corresponding sub-domain. To find a surface for a particular isovalue V, the method starts at the root node and visits all nodes whose extrema interval bracket the isovalue. The T-BON method produces a BONO for each time-step. For each new time-step, all prior geometry is thrown away, and new geometry is produced, as in the normal BONO method. The per-time-step BONO is accessed in a demand-page fashion.

Shen[11] introduced the Temporal Hierarchical Index Tree (THIT)[11] that constructs a "bucket" span space table to accelerate isosurface extraction, where the minimum and maximum values for a cell correspond to the cell's minimum and maximum considering all time-steps. Each table entry contains a temporal subdividing binary tree. The root node records isosurfaces that intersect the cell for all time-steps, intermediate nodes record isosurfaces that intersect the cell for intermediate time intervals, and leaf nodes record isosurfaces that intersect the cell for a single time-step. The algorithm requires that the grid topology remains fixed over time.

Our time-adaptive multi-resolution algorithm exploits the occurrence of small changes and similarities over time in spatial-identical regions. Using error tables that describe the difference of a region at a given resolution and the same region at maximum resolution in comparison to other time-steps, we can determine the exact error introduced by this region. Only when a region does not meet a user-defined error-condition we need to update this region. This approach allows us to reuse as many regions as possible by exploiting the properties of a multi-resolution dataset, where the coarsest representation that meets the error-criterion can be reused.

3. BACKGROUND - MULTI-RESOLUTION TECHNIQUES

Multi-resolution (MR) methods offer the possibility to render a dataset at different resolutions for different subregions of a dataset. Subregions with more detail can be rendered at higher resolution, whereas subregions with less detail can be rendered at lower resolution. This paradigm allows us to render a dataset using a fraction of the original representation without a noticeable loss of information, see Fig. 1. Fig. 1 was produced using a multi-resolution quad-tree decomposition. Each node of the quad-tree contains image data; leaf nodes contain the original image; interior nodes contain approximation images, with the root node containing the coarsest approximation. To produce an MR image, given some error-condition, we start at the root node and evaluate the error for that node. If the error-condition is met, we render that node, otherwise we visit and test the child nodes. We continue this process until the error-condition is met or a leaf node is found (that is rendered). For Fig. 1, the error-condition is the distance from the Mona Lisa's right eye to a node's center. This distance must be greater than the length of the node's diagonal.



(a) (b)

Figure 1. Multi-resolution (MR) representation of a Mona Lisa image. Both (a) and (b) show the same MR image; image (b) shows the outline of the image tiles. All tiles contain the same number of pixels.

4. APPROACH

We extend the idea of MR representation of a dataset to time-varying datasets. Our approach is based on spatial binary subdivision of a time-varying dataset into hierarchically organized regions, down to a maximum depth d_{max}, together with error tables E_R^d for each region R describing the errors over time (to determine per region the error introduced with advancing time). During each render-cycle of a time-step t, the visible regions are checked for their error between the visible representation $R^d(t_v)$ and the correct representation $R^{d_{max}}(t)$, where "visible" means that the region is used for rendering.

Depending on the error of a region it is either reused at time-step t "as is" with the visible representation $R^d(t_v)$, replaced by the representation for the current time-step $R^d(t)$ with $t \neq t_v$, or refined by replacing itself with its subregions R_i^{d+1} to reduce error. Using this error-driven approach our goal is to reduce the number of subregions reloaded per frame, meeting a given error bound where possible.

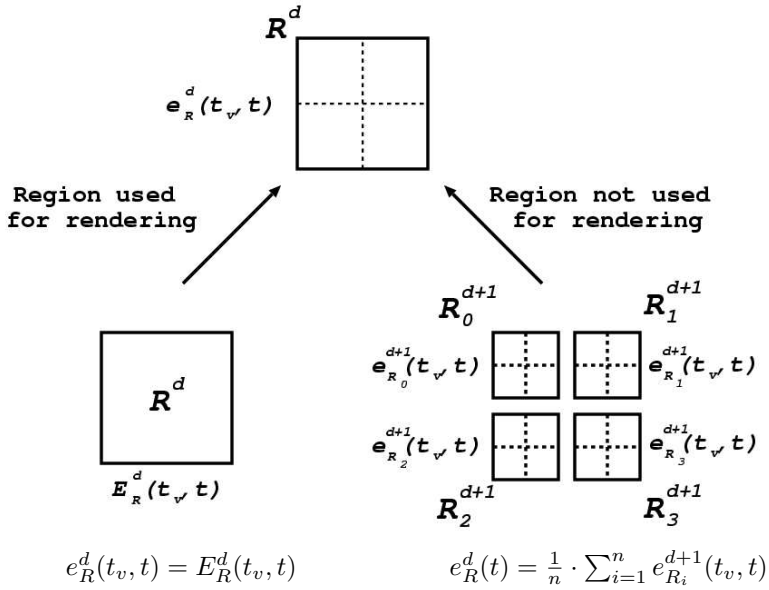
In most simulations, changes within a region are small for small time-steps when compared to larger time-steps. Using a maximum error bound e_{max} and exploiting the similarity of regions between consecutive frames from the visible regions, a set of subregions is determined that must be replaced in order to meet the error requirements. Depending on the current resolution and the change of values over time, a region with minimal activity can be reused for several steps until it no longer meets the error-condition ($R^d(t) \approx R^d(t+k)$, $e_R^d(t, t+k) < e_{max}$ for $k > 0$). The advantage of this approach is that it is independent of the dimension and the representation of the underlying dataset as long as the representation provides multiple resolution levels and error tables.

4.1. Error Tables

The most appropriate way of determining errors $e^d(t_1, t_2)$ between different temporal representations of a region R^d is to use $N \times N$ error look-up tables E_R^d (with N being the number of time-steps). For each region R^d an error table E_R^d is generated that contains the error $e_R^d(t_1, t_2)$ between every time-step t_1 and time-step t_2 of the region at the current depth d in the hierarchy to the corresponding region at the biggest depth d_{max}. The equation $e_R^{d_{max}}(t, t) = 0$ can be guaranteed for leaf-regions R^d with $d = d_{max}$ within the hierarchy.

4.2. Error for a Time-step

Each visible region R_v^d manages the time t_v it represents and the error $e_{R_v}^d(t_v, t)$ it introduces. The error $e_R^d(t_v, t)$ for a region R^d at time t is determined by either the value in the error table when the region itself is displayed (e $= E_R^d(t_v, t)$) or by the average sum of the errors of its subregions R_i^{d+1} by $e_R^d(t) = \frac{1}{n} \sum_{i=1}^{n} e_{R_i}^{d+1}(t_v, t)$, see Fig. 2.



$e^d_R(t_v, t) = E^d_R(t_v, t)$ $\qquad$ $e^d_R(t) = \frac{1}{n} \cdot \sum_{i=1}^{n} e^{d+1}_{R_i}(t_v, t)$

Figure 2. Calculation of error $e^d_R(t_v, t)$ of a region.

4.3. Management of Regions

In order to prevent the re-evaluation of errors for all the regions R within the hierarchy during every time-step, the evaluation is restricted to error table look-up operations for visible regions R^d_v and propagation of updates to their parents only for error re-calculation.

4.4. Replacement Strategy

Our replacement strategy is an extension of the multi-resolution approach for steady datasets to time-varying datasets. A static approach would consider the error of the dataset at the highest resolution, subdividing a region only when the error tolerance is not met. We extend this approach to time-varying datasets by re-using the previous representation where possible. When the previous representation does not meet the error-bound, we use the static hierarchical representation for the current time-step. Using the previous and the current representation, we decide which regions to reuse and which to replace, using the following strategy: We first compare regions that have been active in the previous configuration and are active in the current configuration. We re-use the previous representation for those regions when those meet the error-bound; otherwise, we replace it with the current representation. Next, we check the remaining regions from the previous representation and re-use those that still meet the error-bound. The remaining regions from the previous representation do not meet the current error-bound, so that the representation from the current time-step is used. As a result, we always obtain a multi-resolution representation for the current time-step that meets the given error-bound replacing only those regions that need to be replaced.

When restricting the maximum traversal depth of the hierarchy, it is not always possible to reach the given error bound. Errors are computed for the given resolution at the current depth to the maximum depth in the tree. If we restrict the level d of refinement for a region R^d to a level higher than the base-level d_{max}, the region cannot be refined, although it may have an associated error greater than the given bound. The minimal error possible for such a region, where $d < d_{max}$, is $e^d_R(t_v, t_v)$ with $e^d_R(t_v, t_v) \geq 0$. Depending on the value for e_{max}, $e^d_R(t_v, t_v) \leq e_{max}$ will not always be true.

5. RESULTS

We have applied our time-adaptive multi-resolution approach and standard (MR) approaches to 2D image sequences: a time-dependent modification of the Mona Lisa (see Section 5.3.1) and a simulation of the Richtmyer-Meshkov instability (see Section 5.3.2).

5.1. Approaches Used for Comparison

5.1.1. Replacement Based on Fixed Resolution

The simplest approach is to replace each region of a dataset at every time-step at a given resolution for all regions. This approach has a fixed and well-known number of replacements and a well-known error for every time-step, which is the error between the original resolution and the selected representation level of the dataset. This approach cannot adapt to a given error-bound, unless we increase the resolution and thus the number of regions to be replaced. We compare our approach to different fixed-resolution representations.

5.1.2. Replacement Based on Adaptive Refinement

A more advanced approach is to generate an MR representation for the dataset based on error-terms describing the error for every time-step for each region, reloading all regions from scratch for every time-step. By starting at the lowest resolution and refining only where the given error-bound requires refinement, the result is an MR representation that meets the given error-bound, where possible. This approach adapts to a given error-bound and optimizes the number of regions needed to render the complete dataset.

5.2. Error Definition

For error analysis, we define the error between two greyscale-images as the mean of all absolute pixel-differences:

$$e\big(image(t_i)), image(t_j)\big) = \frac{1}{xsize * ysize} \sum_{x=0}^{x<xsize} \sum_{y=0}^{y<ysize} \Big|\big(image(t_i)(x,y) - image(t_j)(x,y)\big)\Big|.$$

We do not use a view-dependent error, where the influence of the error-term could diminish, for example, with increasing distance from the point of interest.

5.3. Examples

For evaluation purposes we have used the following two datasets: an image of the Mona Lisa perturbed over 100 time-steps, and an image sequence of the Richtmyer-Meshkov instability[12] dataset using 137 images. Both datasets were subdivided into 256 regions at the highest resolution. To evaluate our strategy we have applied our time-adaptive approach, the time-static adaptive refinement approach, and the straightforward replacement approach for different resolutions to both image sequences.

5.3.1. Mona Lisa

For simple evaluation, we have used a snapshot of the Mona Lisa and perturbed the image I according to

$$I'(x,y) = I(x,y) \cdot \big\|(\cos(2^{1-r}) \cdot \Pi\big\|,$$

where

$$r = \begin{cases} 1 & \text{if } r > 1 \\ \sqrt{dx^2 + dy^2} & r \leq 1 \end{cases},$$

with $dx = (x - 0.5 * imageWidth)/(0.5 * imageWidth)$ and $dy = (y - 0.5 * imageHeight)/(0.5 * imageHeight)$.

The image sequence consists of 100 images (snapshots see Fig. 3). Changes within the image occur in the center and propagate to the outer regions of the image in a concentric circular fashion with the number of circles increasing by one circle per frame. Fig. 4 shows several snapshots of the Mona Lisa image sequence, with the regions replaced by our approach shown in white, using an error-bound of 1%. Fig. 5 shows the corresponding graphs.

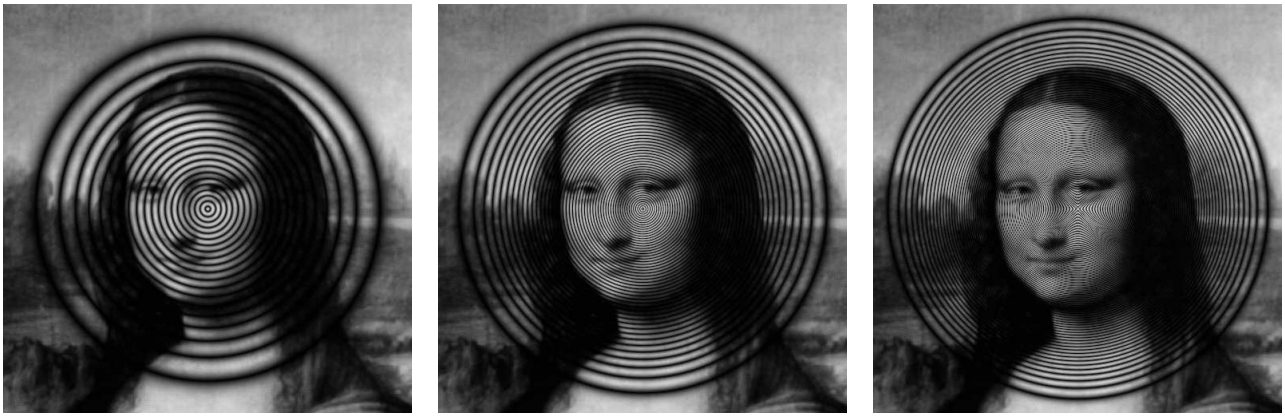


Figure 3. Images from Mona Lisa image sequence; several time-steps.

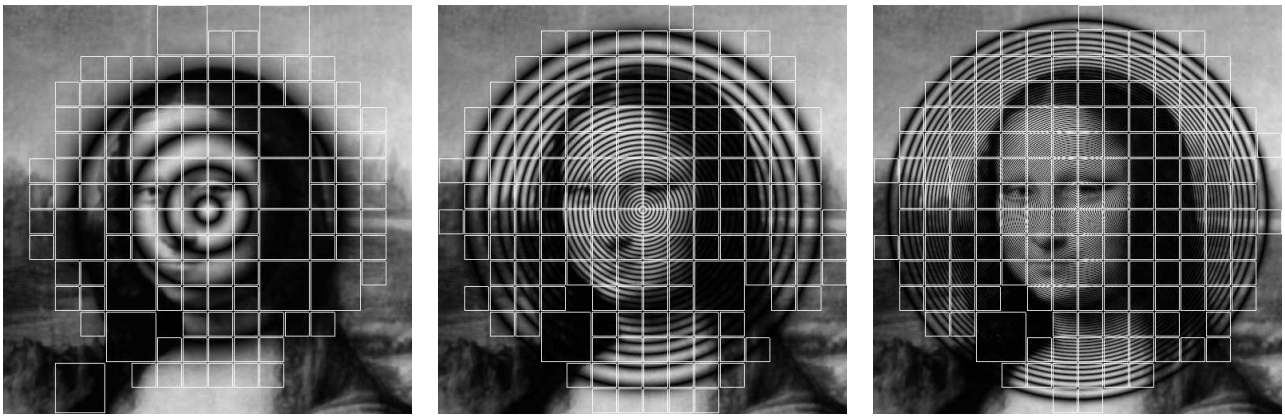


Figure 4. Images from Mona Lisa sequence generated with our approach; regions replaced shown in white; error-bound 1%.

Table 1 summarizes the overall numbers of regions loaded and rendered for each strategy. Our approach loads a smaller number of regions for the complete sequence than the time-static MR approach. The number of regions loaded is smaller for the simple high-resolution replacement-approach (Fig. 5(a)), and the error generated is slightly larger (Fig. 5(b)). This fact is due to the large number of regions of change over time in the Mona Lisa dataset. The straightforward approach performs poorly at lower resolutions due to increasing detail in the inner region of the image. Depending on the fixed number of regions used, the error rises to 4.9% and 6.7% for the Mona Lisa dataset (Fig. 5(b)). Both MR approaches show an average pixel-error of less than 1%, with our approach loading only 69% of the number of regions when compared to the time-static adaptive approach (Table 1, Fig. 5).

Strategy	Mona Lisa
time-adaptive MR	16973/24484
time-static MR	24484
Standard Level 2	1600
Standard Level 3	6400
Standard Level 4	25600

Table 1. Mona Lisa sequence: numbers of all regions loaded/rendered over time; error-bound 1%.

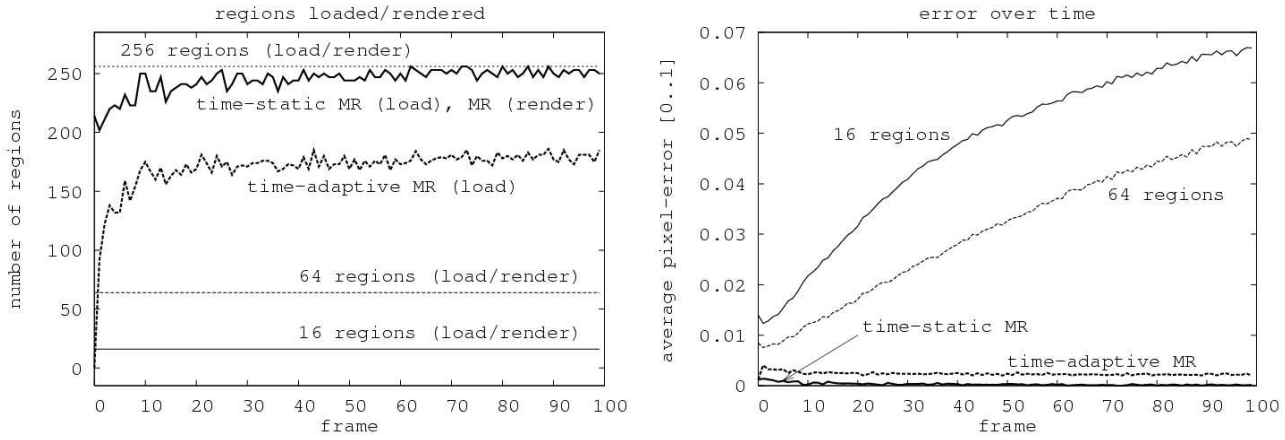


Figure 5. Mona Lisa Sequence: (a) number of regions loaded/rendered; (b) rendered error per frame; error-bound 1%. (Note that the number of regions loaded and rendered is the same for all time-static approaches; the number of regions rendered is the same for the time-static MR approach and our approach; the error using 256 regions is zero.)

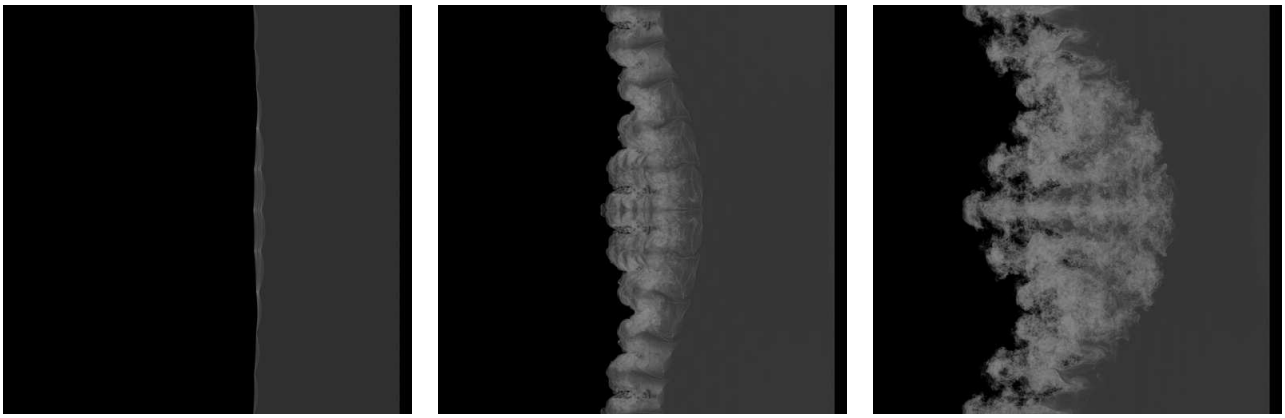


Figure 6. Images from Richtmyer-Meshkov image sequence; several time-steps.

5.3.2. Richtmyer-Meshkov Instability

The Richtmyer-Meshkov instability image sequence is a slice through a 3D-simulated dataset of a Richtmyer-Meshkov instability.[12] The simulation describes the process of two fluids mixing after a shock wave has passed through them. The images show a cross-section perpendicular to the direction of the shock wave. The image sequence consists of 137 images. Fig. 6 shows several snapshots of the original sequence, and Fig. 7 shows snapshots using our approach. Although it seems that in this dataset changes occur only within the center region, there are also significant, but less visible, changes in the direction of the shock wave. The difference between this dataset and the Mona Lisa dataset is that the Mona Lisa dataset has a large region of change, whereas the Richtmyer-Meshkov instability sequence has a region of change that is changing its size and location.

Table 2 summarizes the numbers of regions loaded and rendered for the different strategies with two different error-bounds. As before, our approach loads a smaller number of regions for the complete sequence when compared to the time-static approach, resulting in the same number of regions to be rendered with a slightly larger error.

It can be seen in Fig. 7 that our time-adaptive MR approach mainly replaces data in regions of change, exploiting similarity of different representations of the same region over time for regions with less or no change.

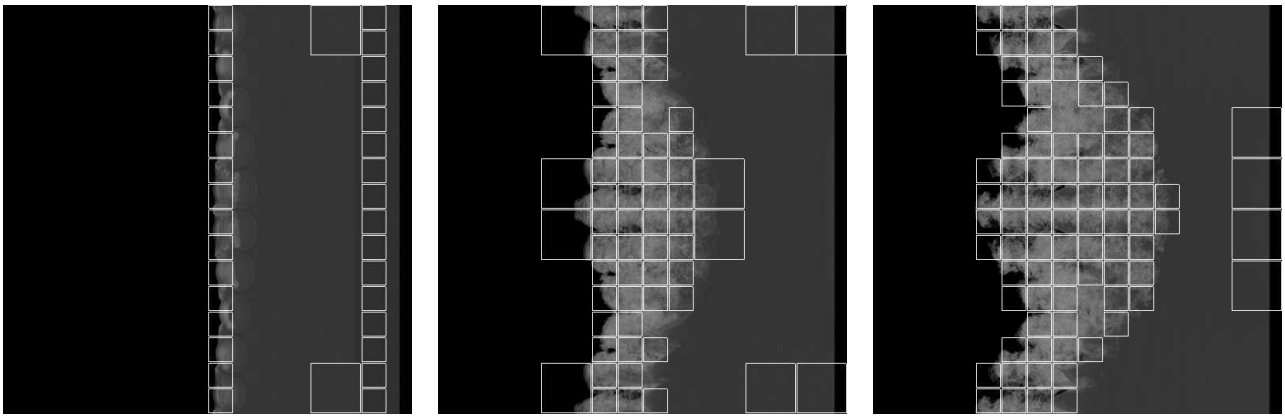
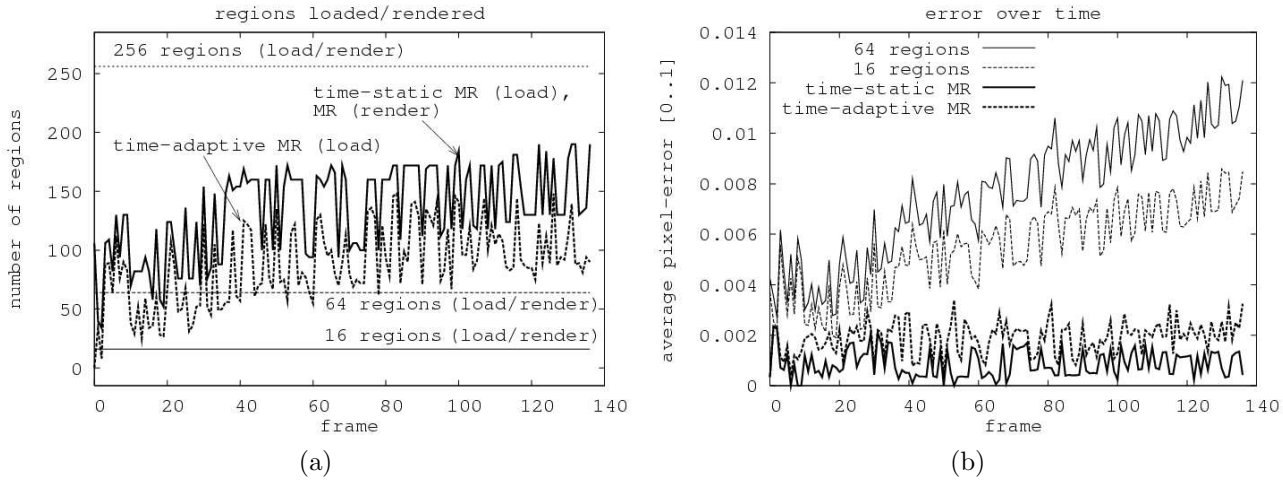


Figure 7. Images from Richtmyer-Meshkov sequence generated with our approach; regions replaced shown in white; error-bound 1%.

Strategy	Richtmyer-Meshkov	
	1%	1.25%
time-adaptive MR	12019/18359	8709/18359
time-static MR	18359	18359
Standard Level 2	2192	
Standard Level 3	8768	
Standard Level 4	35072	

Table 2. Richtmyer-Meshkov sequence: numbers of all regions loaded/rendered over time; error-bounds 1% and 1.25%.



(a) (b)

Figure 8. Richtmyer-Meshkov sequence (error-bound 1%): (a) number of regions loaded/rendered; (b) rendered error per frame. (Note that the number of regions loaded and rendered is the same for all time-static approaches; the number of regions rendered is the same for the time-static MR approach and our approach; the error for 256 regions is zero.)

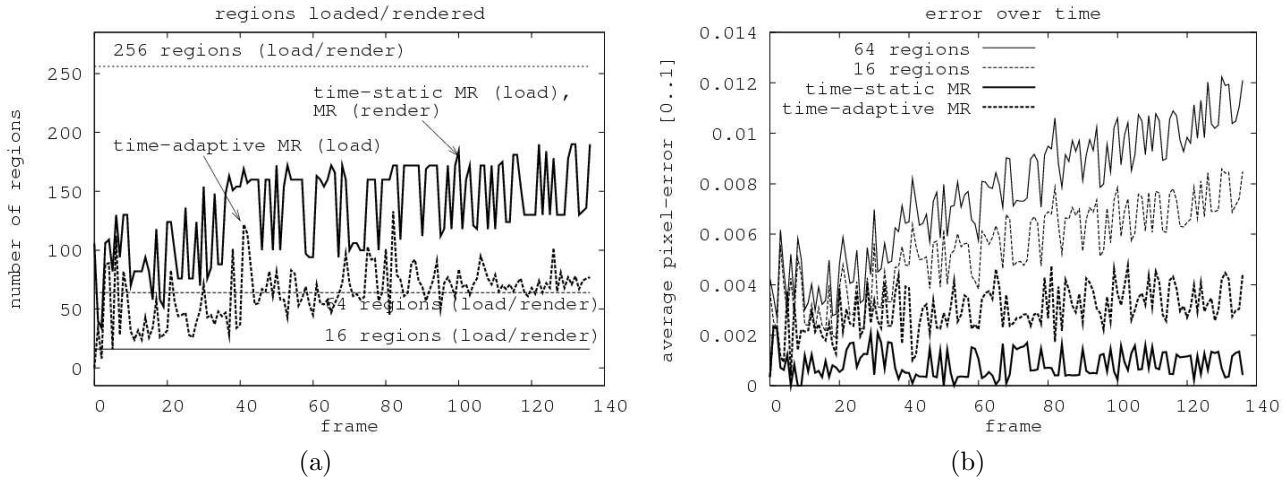


Figure 9. Richtmyer-Meshkov sequence (error-bound 1.25%): (a) number of regions loaded/rendered; (b) rendered error per frame. (Note that the number of regions loaded and rendered is the same for all time-static approaches; the number of regions rendered is the same for the time-static MR approach and our approach; the error for 256 regions is zero.)

The more localized "dynamics" of this image sequence can also be seen in the changing number of regions loaded per time-step (Fig. 8(a)) and the graph showing the error over time of the different approaches (Fig. 8 (b)). The curves representing the error for the fixed-resolution approaches are less smooth when compared to the Mona Lisa sequence. Using an error-bound of 1%, our approach reloads 65% percent of the regions loaded and rendered for a time-static MR approach (Table 2), resulting in an error less than 0.4% (Fig. 8(b)). For an error-bound of 1.25% we load about 47% of the regions rendered (Fig. 9(a), Table 2) with an error less than 0.5% (Fig. 9(b))). While the time-static MR approach loads the same number of regions for both error-bounds, the error with our technique gets slightly larger, but we stay well within the given bounds, loading much less regions.

6. CONCLUSIONS AND FUTURE WORK

The advantage of our approach is that the number of regions replaced is reduced when compared to fixed-resolution methods or time-static MR methods. Compared to the time-static adaptive refinement strategy our algorithm also replaces noticeably fewer regions per time-step without a noticeable increase in visible error, providing better refinement.

The usage of errors is independent of the underlying data representation and the size of the dataset. It allows us to compare datasets of different time-steps and decide which regions to replace, without accessing the dataset itself. The sizes of the error tables depend only on the number of subdivisions and the number of time-steps available, but not on the actual size of the datasets.

When compared to the static adaptive refinement our approach performs well when applied to datasets characterized by large regions with high detail, but hardly any changes over time. The high detail will force the static adaptive approach to refine and reload for every time-step.

We have shown that our approach reduces the amount of data to be reloaded when compared to standard replacement strategies while meeting given error-bounds. Changes in time are not restricted to unidirectional, successive changes - they can be forward and backward for arbitrary distances.

We plan to extend our approach, moving from complete updates to incremental updates on a per-frame basis, which imposes a load and rendering budget, restricting the amount of data to be loaded and rendered per frame. It is also necessary to reduce the complexity of the error tables, which currently has a complexity of $O(N^2)$, causing error tables to grow with an increasing number of time-steps.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award) and ACI 0222909, through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the National Institute of Mental Health and the National Science Foundation under contract NIMH 2 P20 MH60975-06A2 and the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159. We thank the members of the Visualization and Graphics Research Group at the Center for Image Processing and Integrated Computing (CIPIC) and the Center for Applied Scientific Computing (CASC) at Lawrence Livermore National Laboratory.

REFERENCES

1. A. Finkelstein, C. E. Jacobs, and D. H. Salesin, "Multiresolution video," in *Proceedings of SIGGRAPH 1996, Computer Graphics*, **30**, pp. 281–290, ACM Press, (New York), 1996.
2. E. C. LaMar, K. I. Joy, and B. Hamann, "Multi-resolution techniques for interactive hardware texturing-based volume visualization," in *IEEE Visualization '99*, D. Ebert, M. Gross, and B. Hamann, eds., pp. 355–361, IEEE, Computer Society Press, 25-29 Oct. 1999.
3. M. Weiler, R. Westermann, C. Hansen, K. Zimmerman, and T. Ertl, "Level-of-detail volume rendering via 3d textures," in *IEEE Volume Visualization 2000*, T. Ertl, B. Hamann, and A. Varshney, eds., pp. 7–13, IEEE, Oct. 8–13 2000.
4. E. C. LaMar, B. Hamann, and K. I. Joy, *Efficient Error Calculation for Multiresolution Texture-based Volume Visualization*. Springer-Verlag, Heidelberg, Germany, 2002.
5. K.-L. Ma, D. Smith, Y. Ming-Shih, and H.-W. Shen, "Efficient encoding and rendering of time-varying volume data," Technical Report TR-98-22, Institute for Computer Applications in Science and Engineering, NASA, June 1998.
6. H.-W. Shen, L.-J. Chiang, and K.-L. Ma, "A fast volume rendering algorithm for time-varying fields using a time-space partitioning (tsp) tree," in *IEEE Visualization '99*, D. Ebert, M. Gross, and B. Hamann, eds., pp. 371–378, IEEE, Computer Society Press, Oct. 25–29 1999.
7. D. Ellsworth, L.-J. Chiang, and H.-W. Shen, "Accelerating time-varying hardware volume rendering using tsp trees and color-based error metrics," in *IEEE Volume Visualization 2000*, pp. 119–128, 155 (CP), IEEE, Oct. 8–13 2000.
8. P. M. Sutton and C. D. Hansen, "Isosurface extraction in time-varying fields using a temporal branch-on-need tree (t-bon)," in *IEEE Visualization '99*, D. Ebert, M. Gross, and B. Hamann, eds., pp. 147–154, IEEE, Computer Society Press, Oct. 25–29 1999.
9. P. M. Sutton and C. D. Hansen, "Accelerated isosurface extraction in time-varying fields," *IEEE Transactions on Visualization and Computer Graphics* **6**, pp. 97–107, Apr./June 2000.
10. J. Wilhems and A. VanGelder, "Octrees for faster isosurface generation," *ACM Transactions on Graphics* **11**(3), pp. 201–227, 1992.
11. H.-W. Shen, "Isosurface extraction in time-varying fields using a temporal hierarchical index tree," in *IEEE Visualization '98*, D. Ebert, H. Hagen, and H. Rushmeier, eds., pp. 159–166, IEEE, Computer Society Press, Oct. 18–23 1998.
12. A. A. Mirin, R. H. Cohen, B. C. Curtis, W. P. Dannevik, A. M. Dimits, M. A. Duchaeineau, D. E. Eliason, D. R. Schikore, S. E. Anderson, D. H. Porter, P. R. Woodward, L. J. Shieh, and S. W. White, "Very high resolution simulation of compressible turbulence on the IBM-SP system," in *Proceedings of the ACM/IEEE SC99 Conference*, IEEE Computer Society, Nov. 13–19 1999.