Component-Based Isosurface Extraction for Multiple Dataset Visualization

Shiyuan Gu
Department of Computer Science,
Department of Mathematics,
Louisiana State University,
USA (70803), Baton Rouge, LA
gshy@math.lsu.edu

Bijaya B. Karki
Department of Computer Science,
Louisiana State University,
USA (70803), Baton Rouge, LA
karki@csc.lsu.edu

ABSTRACT

In the situations where isosurfaces are comprised of many disjoint components, two or more datasets can be visualized simultaneously by processing only a subset of isosurface components. The components of interest can be selected by exploiting interdataset coherency at the level of individual voxels and components. Thus, only those components (identified as voxel coverages or voxel sets) which differ significantly among the datasets under consideration are extracted as needed while the similar components were extracted only once from a reference dataset. Since the polygons are extracted/rendered as a whole component, the rendered isosurfaces are crack-free. We use three user-defined thresholds to control multiple dataset visualization (MDV) so that important relationships (differences and similarities) among the datasets can be explored with an improvement in the overall performance. If the data-coherency can not be defined easily, MDV can still benefit from the on-the-fly processing of the individual components.

Keywords: Multiple dataset visualization, data coherency, isosurface extraction, component, crack-free

1 INTRODUCTION

Multiple dataset visualization (MDV) is desirable for developing a more complete understanding of a given system or problem for which two or more datasets are available (e.g., [aoy07, chi97, kha06]). Unlike normal visualization in which each dataset is processed independent of other dataset(s), MDV processes multiple datasets of interest together so that the cross-correlations (such as differences and similarities) among them can be better explored. A straightforward approach is to completely process each dataset using an existing visualization technique (such as isosurface extraction [lor87]) and then simply display the resulting images together. There are two issues with this approach. Firstly, it is not always possible to process fast and completely all the datasets under consideration due to the limited computing and storing capabilities. For instance, a modern dataset can be arbitrarily large so minimizing the processing/storage requirements is crucial in MDV. Secondly, even if it is possible to handle all datasets simultaneously, it may not be effective for comparison purpose. For instance, there are situations where the underlying structures are highly complex and widely spread in a three-dimensional space. The users often face tremendous challenge in visually identifying the similar and dissimilar regions. It is, therefore, desirable that MDV can automatically detect and highlight such regions.

Several approaches have been explored in the context of visualization of multiple datasets. In [chi97], the authors have proposed visualization spreadsheets for comparing multiple datasets. By displaying the outputs for several datasets in a tabular form, the users can apply various operations for the cells of the spreadsheets (e.g., subtracting two cells and rendering the difference in another cell). A recent MDV work involves overlaying multiple visualizations within a single view or rendering them in multiple views [aoy07]. While these approaches primarily deal with the interfaces (i.e., how the display outputs are organized), the scalable adaptive MDV presented in [kha06, kha07] deals with visualization at the processing level using isosurface extraction and texture mapping.

To improve MDV performance, one can exploit the coherency between the datasets to be visualized ([kha08]). Data coherency is simply a measure of similarity between the datasets. In this approach, a given multiple set of data are first divided into two groups: reference datasets and non-reference datasets. The reference datasets are those which are completely processed and whose polygon data are used subsequently to also represent the parts of the isosurfaces of the non-reference datasets. The required data-coherency test is performed by comparing a non-reference dataset with a reference dataset block

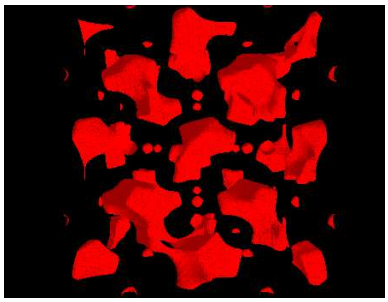


Figure 1: Multicomponent isosurface

by block. Only the polygons (triangles) within the blocks (i.e., octree nodes) which significantly differ from the corresponding blocks of the reference dataset are extracted; the polygons in other blocks are simply retrieved from the reference dataset.

In this paper, we perform component-based isosurface extraction in combination with interdataset coherency to support multiple dataset visualization. Our MDV is shown to be useful in the situations where the underlying isosurface structure is comprised of many small spatially disjoint components. An example is shown in Fig. 1 for the electron density isosurfaces. Here, different isosurface components correspond to electron distributions around different atomic sites. Similarly, in the case of medical data such components may correspond to different suborgans or tissue structures. A typical scenario can be that two or more datasets represent some changes which are localized. For instance, one vacancy site (an missing atom) in a crystal is likely to affect the electron distribution only in its neighbourhood. A component can thus change its size/shape freely without affecting the other components. Therefore, identifying and subsequently processing such individual components can be useful in comparing datasets and hence in MDV. As stated above, our method assumes the data coherency, i.e., it assumes that the shape differences between isosurfaces are reflected in a subset of voxels. It differs from the data coherency method of [kha08] in that we analyze both the component similarity and voxel similarity while only the voxel similarity was considered previously. There are two advantages of considering both types of differences. First, the isosurfaces are guaranteed to be crack-free while the previous study ([kha08]) suffers from cracks due to the inconsistency between extracted and approximated parts of isosurfaces (Fig. 2). Secondly, the proposed approach is more effective in identifying the interesting structural differences and suppressing the noises. In many cases, a small change in a big isosurface component may not be of interest because it can be due to some noise or can be so small that it is visually undetectable. We can avoid an unnecessary processing of the corresponding polygons by specifying appropriate thresholds.

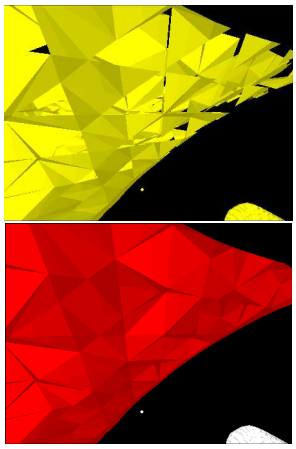


Figure 2: Cracks (top) and no cracks (bottom)

2 MULTI-COMPONENT ISOSURFACE EXTRACTION

The first important step in the proposed multi-component isosurface extraction method is to identify the components themselves. An exact identification of a component can be computationally expensive. However, we can approximate an isosurface component by its voxel coverage - a set of voxels which contribute to the component, which is determined as described below. It is easy to realize that two components must be separated spatially from each other by, at least, one voxel gap. The second important step is to perform the interdataset coherency test. For this purpose, we divide given datasets into two categories: reference datasets (*RDS*) and non-reference datasets (*NRDS*). Consider the simplest case of two datasets for which we have one *RDS* and one *NRDS*. *RDS* is completely processed and the polygon data are stored for later retrieval. On the other hand, *NRDS* is processed for partial isosurface extraction. Only those components which differ from or do not exist in *RDS* are directly extracted from *NRDS*; all similar components are rendered using the corresponding *RDS* polygon data. Thus, the two types of datasets are processed differently.

The *RDS* isosurface extraction is performed using Algorithm 1. This algorithm processes a reference dataset to generate the polygon data representing isosurface component and a lookup table $RDS.map$ representing the relationship between voxels and the components. An entry $RDS.map[v_i]$ indicates the component the voxel v_i belongs to. We define $RDS.map[v_i]$ to be zero if v_i does not contribute to the isosurface. $RDS.map$ is used later during processing of *NRDS* (Algorithm 2). $RDS.map$ allows us to easily find out whether a newly extracted component in *NRDS* overlaps with any component in *RDS*. In Algorithm 1, $v_i.iss$ and $v_i.p$ are true/false flags indicating whether the voxel v_i contributes to the isosurface and whether

the polygons within it have been extracted, respectively. These two flags allow us to skip many voxels. The variable *comp* counts the number of components. To find a component, we start from a voxel in that component and expand the component by checking its neighbors and then their neighbors successively. This expanding process can be implemented by a queue as follows. We traverse each voxel of *RDS*. If a voxel contributes to an unexplored component, we extract the polygons within the voxel and check for its neighbors. We put all its neighbors that also contribute to the isosurface into a queue (denoted as Q). We repeat this process for each member in the queue until all members in the queue are processed. We record the component a voxel belongs to in *RDS.map* every time polygons are extracted from the voxel. We also group the polygons in a component together for an efficient retrieval. Note that since the polygons that belong to the same component are extracted successively, the grouping can be done simply by recording the first extracted polygon in the component if the polygons are saved in a list.

We use the Marching Cubes algorithm to extract the polygons in a voxel. In order to determine whether a voxel contributes to the isosurface or not, we need to compare its scalar values to a given isovalue. It is not a cheap process. One can find the maximum and minimum of the scalar values of each voxel and use the octrees to speed up the extraction. Here, our major goal is to illustrate the component-based isosurface extraction approach. It can be extended to octrees ([she96, wil92]) and other fast isosurface schemes ([yar98]).

For each non-reference dataset, we first compute the per-voxel differences from a reference dataset. The voxel differences between two datasets can be measured in many ways. Different measures vary in their sensitivity to noise and their ability to capture the change. Generally speaking, the measures more capable to capture the change are also more sensitive to noise. Some choices include the differences in one of the eight values or the maximum or the average value between two voxels. Computing the voxel difference can be time-consuming. However, if we choose a difference measure independent of isovalue such as those just mentioned, we need not recompute the differences when we change the isovalue. It is important to note that isosurfaces need to be explored over a wide range of isovalues. We can also divide the whole volume into blocks and assign the same voxel difference to all voxels in the same block as in the previous study ([kha08]).

Once the voxel differences are computed, the polygon generation for a non-reference dataset can be performed with Algorithm 2 only for the components which contain sufficiently large number of significant

Input: a reference dataset *RDS*
Output: polygons in *RDS*, *RDS.map*

Initialize all variables to zeros/false.
FOR each voxel v_i in *RDS*
  IF $v_i.iss$ AND NOT $v_i.p$ **THEN**
    extract polygons in v_i
    $v_i.p \leftarrow true$
    $comp \leftarrow comp + 1$
    $RDS.map[v_i] \leftarrow comp$
    $Q[0] \leftarrow v_i$
    $head \leftarrow 0$, $tail \leftarrow 1$
    WHILE $head < tail$
      FOR each neighbor nb_j of Q[head]
        IF $nb_j.iss$ AND NOT $nb_j.p$ **THEN**
          extract polygons in nb_j
          $nb_j.p \leftarrow true$
          $RDS.map[nb_j] \leftarrow comp$
          $Q[tail] \leftarrow nb_j$
          $tail \leftarrow tail + 1$
        ENDIF
      ENDFOR
      $head \leftarrow head + 1$
    ENDWHILE
  ENDIF
ENDFOR

Algorithm 1: Pseudo-code for isosurface extraction for a reference dataset

voxels. These are the voxels whose difference are greater than the user-defined $threshold1$. In Algorithm 2, $v_i.p$ is a true/false flag which indicates whether the voxel has ever entered into the queue Q or not. We use the flag $v_i.p$ to avoid putting the same voxel into Q multiple times. $voxel.diff$ denotes the voxel difference, $threshold1$, $threshold2$, $threshold3$ are user-defined parameters. $threshold2$ and $threshold3$ are for measuring the differences between components while $threshold1$ is for measuring the similarity between voxels. nV is a counter counting the number of voxels in the component. nSV is a counter counting the number of significant voxels in the component. nV and nSV together with $threshold2$ is for determining whether a component is significantly different and hence should be extracted. $NE[.]$ is an array of true/false flags and $NE[c_i]$ indicates whether there are newly extracted polygons in the corresponding region of $RDS.c_i$ (the c_i component in *RDS*). $NE[.]$ is used when we decide the components to be retrieved from *RDS*.

Algorithm 2 scans through all significant voxels and finds their components. The polygons in a component are extracted only if the component contains sufficiently large number (determined by $threshold2$) of significant voxels. The polygons of the reference dataset are retrieved if no polygons are

newly extracted in the corresponding region in the non-reference dataset and the overlap of the *NRDS* isosurface and *RDS* isosurface are sufficiently large (determined by $threshold3$). The effects of these three thresholds are presented in the next section.

Input: a non-reference dataset *NRDS*, polygons in the reference dataset *RDS*, *RDS.map*, $threshold1$, $threshold2$, $threshold3$.
Output: polygons in *NRDS*

Initialize all variables to zeros/false
FOR each voxel v_i in *NRDS*
  IF $v_i.iss$ AND NOT $v_i.p$ AND $v_i.diff > threshold1$ **THEN**
    $v_i.p \leftarrow true$
    $nV \leftarrow 1$; $nSV \leftarrow 1$
    $Q[0] \leftarrow v_i$
    $head \leftarrow 0$; $tail \leftarrow 1$
    WHILE $head < tail$
      FOR each neighbor nb_j of Q[head]
        IF $nb_j.iss$ AND NOT $nb_j.p$ **THEN**
          $nV \leftarrow nV + 1$
          IF $nb_j.diff \geq threshold1$ **THEN**
            $nSV \leftarrow nSV + 1$
          ENDIF
          $nb_j.p \leftarrow true$
          $Q[tail] \leftarrow nb_j$
          $tail \leftarrow tail + 1$
        ENDIF
      ENDFOR
      $head \leftarrow head + 1$
    ENDWHILE
    IF $nSV > threshold2 \times nV$ **THEN**
      FOR $k = 0, \ldots tail - 1$
        extract polygons in $Q[k]$
        $NE[RDS.map[Q[k]]] \leftarrow true$
      ENDFOR
    ENDIF
  ENDIF
ENDFOR
FOR each component $c_i (i \geq 1)$ in *RDS*
  IF NOT $NE[c_i]$ **THEN**
    $n1 \leftarrow$ number of voxels in c_i
    $n2 \leftarrow$ number of voxels in c_i whose counterparts in *NRDS* also contain polygons.
    IF $n2 > threshold3 \times n1$ **THEN**
      retrieve the polygons in c_i
    ENDIF
  ENDIF
ENDFOR

Algorithm 2: Pseudo-code for isosurface extraction for a non-reference dataset

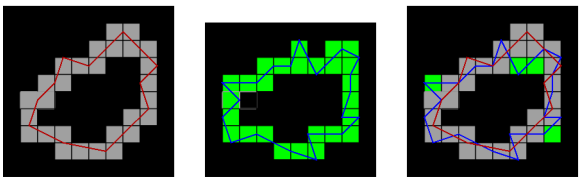
3 THRESHOLDS AND THEIR SIGNIFICANCE

In this section, we analyze how thresholds control the outputs. We consider a two-dimensional case but our discussion and all conclusions apply to a three-dimensional case. Imagine that the components from the reference dataset and the non-reference dataset are laid over each other in space (Fig. 3). We color the *RDS* isosurface (isoline) blue and the *NRDS* isosurface (isoline) red. There are three cases for each component in the output for a non-reference dataset:

- Case I: One component from the non-reference dataset does not touch any component from the reference dataset (Fig. 3 (a)). If the voxel covering of the component contains more than $threshold2$ significant voxels, the polygons from the non-reference dataset are extracted. Otherwise, the polygons are missing in the output.
- Case II: One component from the reference dataset does not touch any component from the non-reference dataset (Fig. 3 (b)). This polygons are not retrieved and hence is missing in the output.
- Case III: One component from the reference dataset overlaps another component from the non-reference dataset (Fig. 3 (c)).
  - Case III (i): The component from the non-reference dataset contains more than $threshold2$ portion of significant voxels. The polygons contained in that component from the non-reference dataset (i.e. the red one) will be extracted. The polygons from the reference dataset (i.e., the blue one) are not retrieved and are missing in the output.
  - Case III (ii): The component from the reference dataset contains no more than $threshold2$ portion of significant voxels and the overlapping of these two components is more than $threshold3$ portion of the total number of the voxels in the component from the reference dataset (In other words, the green voxels in Fig. 3 (c) count less than $1 - threshold3$ portion of the number of the voxels in the component from the reference dataset). The polygons from the reference dataset (the blue one) are retrieved and the polygons from the non-reference dataset are missing in the output. In other words, the exact isosurface of the non-reference dataset (the red one) in that region is approximated by the isosurface of the reference dataset (the blue one).

– Case III (iii): Otherwise, no polygons are extracted or retrieved for the non-reference dataset.

Here, *threshold*1 is for capturing the difference locally while the *threshold*2 and *threshold*3 are for capturing the global difference (between components). By choosing different combinations of three thresholds, users can define the similarity and suppress noise in a flexible way. If all three thresholds are set zero, the output are the exact isosurfaces from the non-reference dataset. From the above analysis, we can also see that we do not introduce any cracks or any self-intersection to the output isosurfaces since the polygons are extracted or retrieved as a whole component.



(a) Case I (b) Case II (c) Case III

Figure 3: Effect of thresholds. The blue and red curves represent an exact component of the isosurface from the reference dataset and the non-reference dataset, respectively

4 PERFORMANCE ANALYSIS

We visualize the electron density data for the perfect and defective (one atom missing) crystals of MgO with the multicomponent isosurface extraction method. We use the perfect crystal dataset as the reference dataset and the defective crystal dataset as the non-reference dataset. The size of each dataset is $356 \times 380 \times 252$. Performance measurements were carried out in Windows XP with Pentium (R)4 3.40 GHz and 2.0 GB RAM.

We use the difference between the scalar values at the upper left corners of the two voxels as the voxel difference. We calculate the number of triangles and polygon generation time by varying *threshold*1 for *threshold*2 and *threshold*3 fixed at 0.25 and compare isosurfaces for four isovalues of 0.01, 0.018, 0.02 and 0.023. As shown in Fig. 4, the number of triangles for the exact *NRDS* isosurface (which are directly processed) varies strongly with isovalue. Correspondingly, the isosurface extraction time increases with isovalue. Note that the time (350 *ms*) for the one-time computation of voxel differences is not included. By choosing *threhold*1, *threshold*2 and *threshold*3 to be 0.0015, 0.25 and 0.25, respectively, we successfully capture the dissimilar components by much fewer triangles in about half of the time. The exact and approximated isosurfaces for *NRDS* shown in Fig. 5 are visually indistinguishable. The different components are highlighted so we can see easily that some disjoint components in the perfect crystal merge together in the defective crystal. As *threshold*1 decreases, more and more voxels in the non-reference dataset turn into significant voxels. However, in our approach more significant voxels do not necessarily lead to more extracted triangles (which is the case in the previous method in [kha08]) because we also consider the similarity between components. If we increase *threshold*2, some components may be discarded so less overhead in polygon extraction and the performance improves. *threshold*3 does not involve polygon extraction but only retrieval for rendering and so it will not affect performance dramatically. Domain-specific knowledge is needed to tune these values. However, there are some visual indications in the case when the threshold values are not set correctly. For example, large blank space of the resulting image may indicate that *threshold*2 or *threshold*3 is set too high. We may need to do further investigation for that particular portion of data.

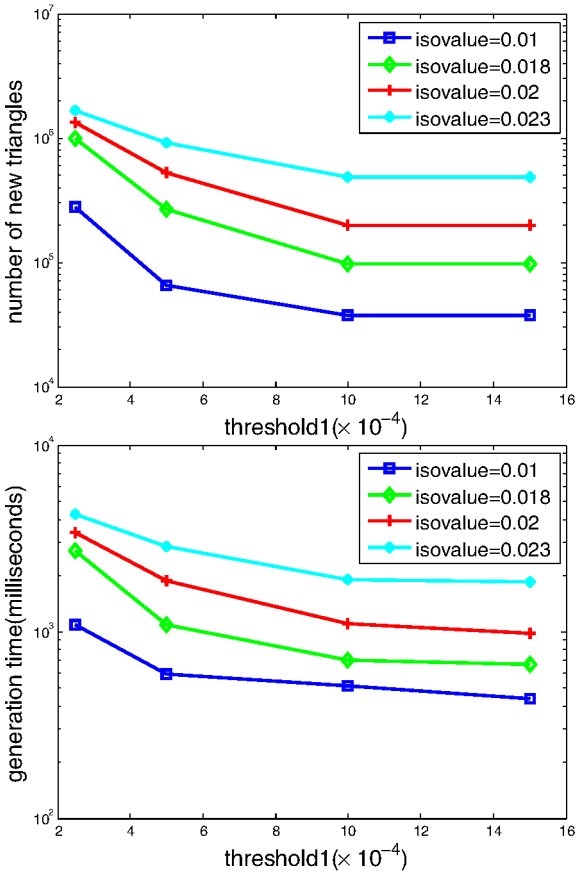


Figure 4: Number of new triangles (top) and generation time (bottom) for the non-reference dataset as a function of *threshold*1

Component-based isosurface extraction is not useful if the isosurfaces under consideration contain only

one or two large components that extend over almost the entire scene. If this is the case, then these big components are either extracted or ignored based on the thresholds. If the components are not extracted, then large portions of the isosurfaces are retrieved or missing. If they are extracted, it takes more time than the normal processing time.

5 ON-THE-FLY ISOSURFACE EXTRACTION

Our method assumes the data coherency. In some situations, the shape similarity/difference may not be reflected by voxel values. Consider the case where two isosurfaces differ only by a shift of one voxel size. The shapes of the isosurfaces are the same but the voxels differ a lot. Here, we provide a way to handle such situations to take the advantage of multi-component isosurface extraction. Instead of extracting the polygons at once, we use a simple shape to represent each component after we find them using Algorithm 1. If the user recognizes and selects a region of interest, then the actual isosurfaces within that regions are exacted. We show an example in Fig. 6 where we use a bounding box to represent a component. Since only simple shapes are stored and the exact isosurfaces are extracted on the fly, we not only improve the time and space but also the user interaction.

6 CONCLUSIONS

In this paper, we have proposed a combination of multi-component isosurface extraction and data coherency methods to support multiple dataset visualization. Our approach can detect and highlight interesting structural differences. By exploring the voxel similarity and component similarity between datasets, we can improve time and space requirements in the situations where only a subset of isosurface components differ significantly between the datasets. We also provide on-the-fly isosurface extraction based on component comparison.

ACKNOWLEDGEMENTS

This work is supported by the NSF Career (EAR 0347204) and NSF/EPSCoR (EPS-0701491) grants.

REFERENCES

[aoy07] Aoyama, D.A., Hsiao, J.-T. T., Rdenas, A.F.C., and Pon, R.K. TimeLine and visualization of multiple-data sets and visualization querying challenge. Journal of Visual Languages and Computing & Computing, vol. 18, 2007

[chi97] Chi, E.H., Riedl, J., Barry, P., and. Konstan, J. Principles for information visualization spreadsheets. Proceedings of the 1997 Information Visualization Symposium.

[han05] Hansen, C.D., and Johnson, C.R. The Visualization Handbook. Elsevier Academic Press, 2005.

[kha06] Khanduja, G., and Karki, B.B. Multiple datasets visualization with isosurface extraction. Proceeding (541) Visualization, Imaging, and Image Processing, 2006

[kha07] Khanduja, G., and Karki, B.B. Using graphics hardware for multiple datasets visualization. WCSG07 Proceedings (Ed. 2007, V. Skala), pp. 161-168, ISBN 978-80-86943-02-2

[kha08] Khanduja, G., and Karki, B.B. Exploiting data coherency in multiple dataset visualization. Proc. of the 10th Int' l. Conf. on Computer Graphics and Imaging (CGIM' 08), 20

[lew03] Lewiner,T., Lopes, H., Vieira, A.W. and Vieira, G. Efficient implementation of Marching Cubes' cases with topological guarantees. Journal of Graphics Tools, 2003.

[lor87] Lorensen, W.E., and Cline, H.E. Marching cubes: a high resolution 3D surface construction algorithm. ACM Computer Graphics, vol. 21, 1987

[nub03] Nubera, E., LaMarb, E. C., Hamanna, B., and Joya, K.I. Approximation of time-varing multiresolution data using temporal-spatial reuse. Visualization and Data Analysis, 2003

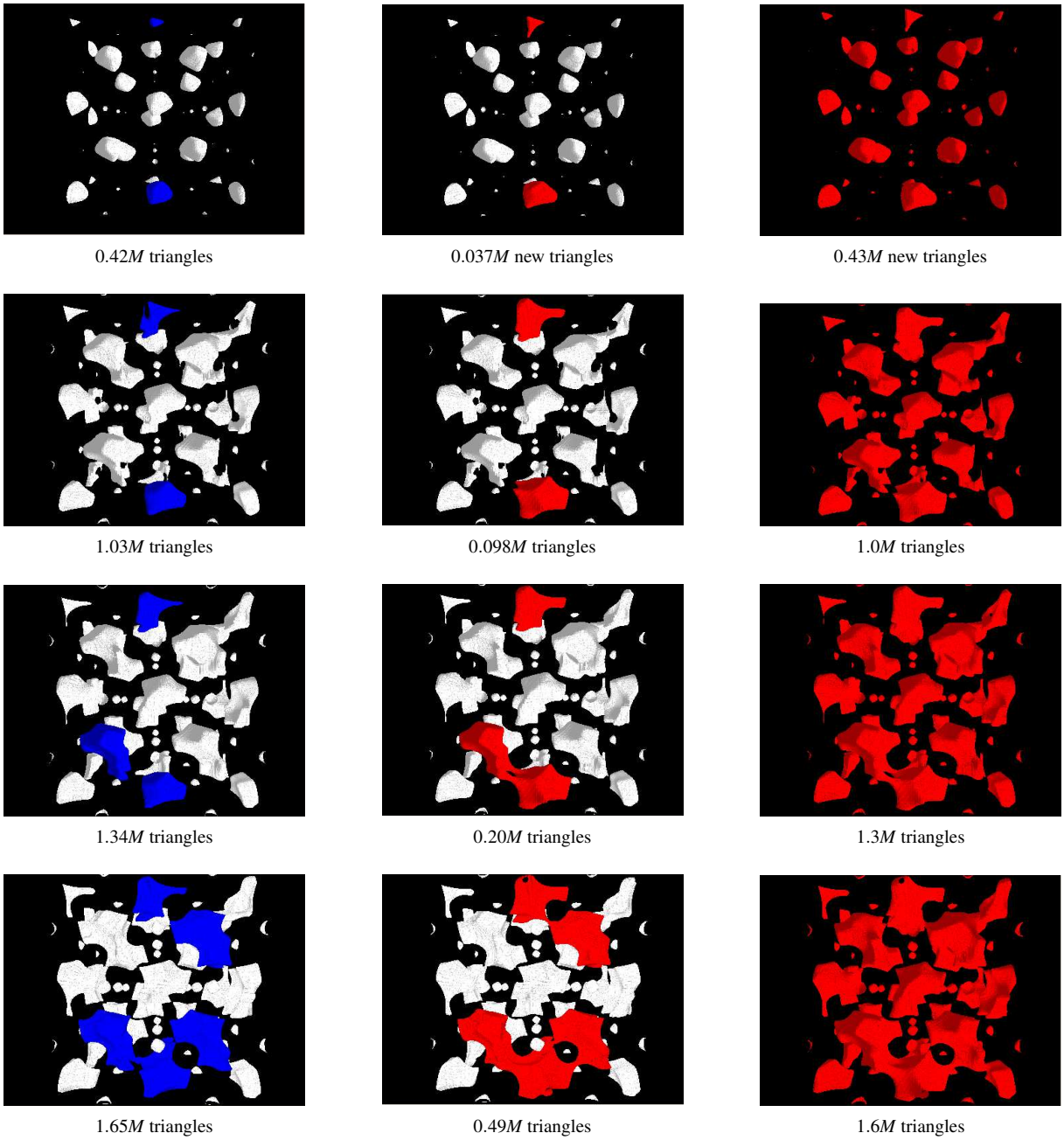
[she96] Shekhar, R., Fayyad, E., Yagel, R., and Cornhill, J.F. Octree-based decimation of marching cubes surfaces, IEEE Visualization, Proceedings of the 7th conference on Visualization 1996.

[wil92] Wilhems, J. and Gelder, A.V. Octrees for faster isosurface generation. ICM Trans. Graphics, Vol. 11, No. 3, pp. 201-227, July 1992.

[yar98] Livnat, Y., and Hansen, C. View Dependent Isosurface Extraction. IEEE Visualization, 1998.

[web01] Weber, G.H., Kreylos, O., Ligocki, T.J., Shalf, J.M., Hagen, H., Hamann, B., and Joy, K.I. Extraction of crack-free isosurfaces from adaptive mesh refinement data. Approximation and Geometrical Methods for Scientific Visualization, pp.19-40, 2001

[wes99] Westermann, R., Kobbelt, L., and Ertl, T. Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. The Visual Computer, vol. 15, pp. 100-111, 1999.



0.42M triangles | 0.037M new triangles | 0.43M new triangles

1.03M triangles | 0.098M triangles | 1.0M triangles

1.34M triangles | 0.20M triangles | 1.3M triangles

1.65M triangles | 0.49M triangles | 1.6M triangles

Figure 5: The first column are the isosurfaces for the reference dataset. The second column are the approximated isosurfaces for the non-reference dataset. The third column are the exact isosurfaces for the non-reference dataset. The blue components are the components not retrieved and hence are unique to the reference dataset. The red components are extracted exactly from the non-reference dataset. The white components are recognized as common components by our approach. The rows are for the isovalue $0.01, 0.018, 0.02, 0.023$ respectively. We can easily see that some disjoint components from the reference dataset merge together to form a new component in the structure from the non-reference dataset. Those components are successfully identified by our approach.

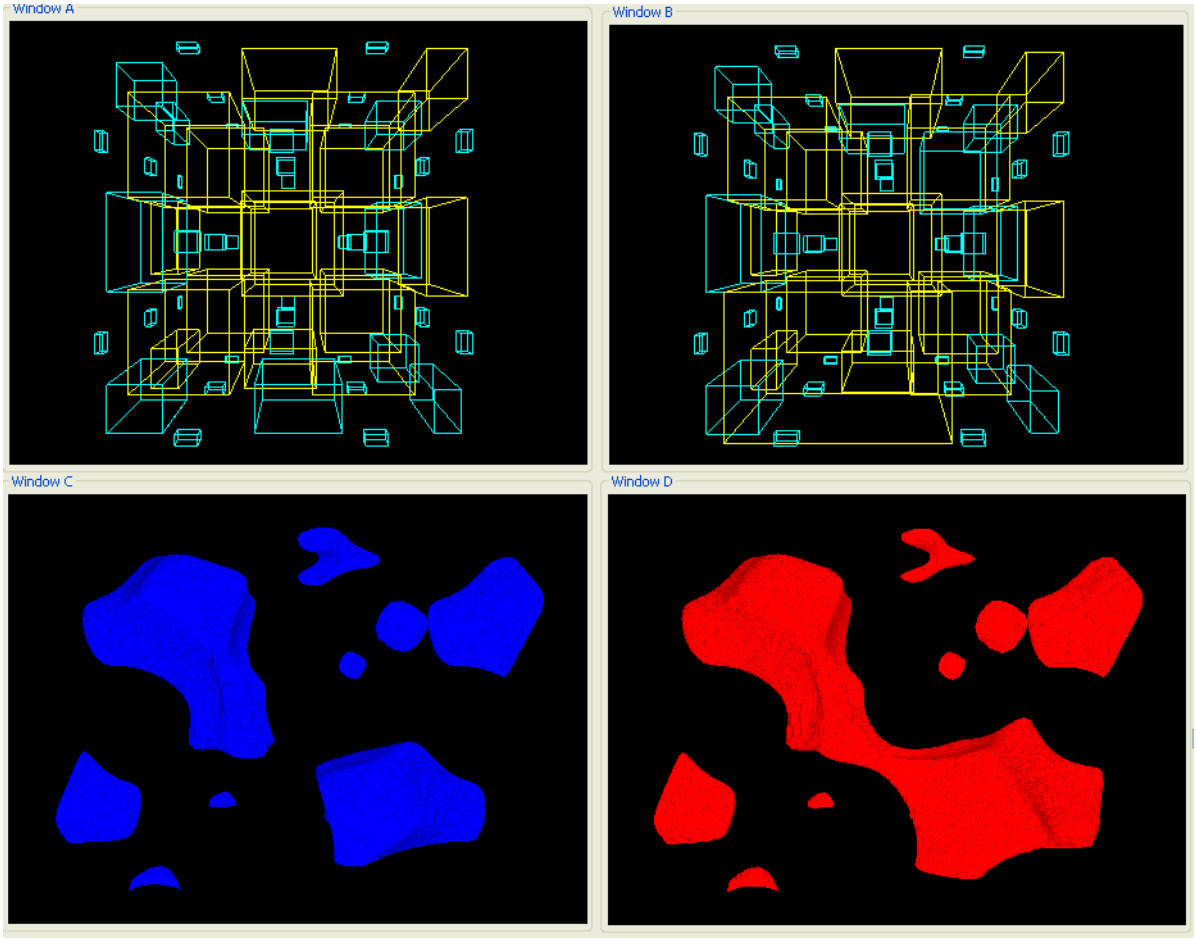


Figure 6: On-the-fly isosurface extraction. Each component is represented as its bounding box. The boxes are colored according to their size. A user recognizes a region of interest by noticing a big yellow box unique to one dataset (i.e., the big yellow box in the bottom left of the top right window) and selects the region so that the isosurfaces within that region are extracted from both datasets.