A New Approach of Seed-Set Finding for Iso-Surface Extraction

Chiang-Han Hung Chuan-kai Yang

National Taiwan University of Science and Technology *

Abstract

Iso-surface extraction is one of the most important approaches for volume rendering, and iso-contouring is one of the most effective methods for iso-surface extraction. Unlike most other methods having their search domain to be the whole dataset, iso-contouring does its search only on a relatively small subset of the original dataset. This subset, called a seed-set, has the property that every iso-surface must intersect with it, and it could be built at the preprocessing time. When an iso-value is given at the run time, iso-contouring algorithm starts from the intersected cells in the seed-set, and gradually propagates to form the whole iso-surface. As smaller seed-sets offer less cell searching time, most existing iso-contouring algorithms concentrates on how to identify an optimal seed-set. In this paper, we propose a new and efficient approach for seed-set construction. This presented algorithm could reduce the size of the generated seed-sets by up to one order of magnitude, compared to the volume thinning approach.

1 Introduction

Volume rendering has been an important research topic in recent years due to its wide applications in various areas, including medical diagnosis, numerical simulations, production of education or entertainment, and so on. While there are numerous techniques of volume rendering, iso-surface extraction is one of the most popular approaches. In general, Iso-surface extraction consists of two phases, cell searching and triangle generation. As the procedure of the second phase is nearly fixed, most of the current research concentrates on reducing the time spent during the first phase, and among such, iso-contouring is one of the most effective methods. The idea of iso-contouring is to first identify a subset called seed-set, which has the property that every iso-surface must intersect with at least one cell from the seed-set. Assuming continuous variation over the scalar fields defined on the cells, iso-contouring algorithm propagates from the intersected cells in the seed-set to form the desired iso-surface. Compared with other iso-surface extraction methods, the benefit of iso-contouring is two-fold. First, the generated iso-surface could be readily converted into triangle strips, which significantly reduces the traffic sent to the graphics card during the rendering stage, thus speeding up the performance. Second, the cell search domain is often dramatically decreased, and therefore the search time for finding the cells on the iso-surface is also minimized. Furthermore, many other techniques that help reduce the cell search time could also be applied, such as *interval trees*, leading to even better overall performance. Compared with the original dataset, the seed-set size should be relatively smaller so that we could quickly locate where to start the iso-surface propagation. This defines the very goal during the preprocessing time of an iso-contouring algorithm. Inspired by the *min-max span space* proposed by [LSJ96], we observed that a seed-set of a dataset could be constructed in a brand-new way, which will be explained in the ensuing sections. This observation leads to a fairly simple implementation with high efficiency. In addition, our proposed method is independent of other optimization techniques, such as the ideas proposed by Bajaj et al. [BPS96, BpS97b, BpS97a], in the sense that it could be applied together with other approaches. For example, it can be shown that when combined with the existing volume thinning approach, the size of the resulting seed-set from the original volume thinning approach could be further reduced to be 8 times smaller than the one using volume thinning approach alone.

The rest of the paper is organized as follows. Section 2 reviews the related work on iso-surface extraction, while section 3 details the technique of volume thinning, which serves as a comparison and test-bed for our proposed approaches, and the concept of min-max span space, whose representation of a dataset inspires our new idea of seed-set construction. Section 4 presents our new algorithms, which could be viewed as new ways for minimizing a seed-set. Section 5 demonstrates the efficiency of our method when compared and/or combined with the volume thinning approach. Section 6 concludes the paper and envisions the potential future directions.

2 Related Work

Volume rendering techniques can be classified into two big categories: direct volume rendering, such as raycasting [Lev88, Lev90], and indirect rendering, such as iso-surface extraction. Lorenson's marching cubes [LC87] pioneered the research on iso-surface extraction. Unlike direct rendering to generate images directly from the datasets, indirect rendering outputs polygonal meshes which are sent to the graphics engines for traditional rendering. In general, given an iso-value, the way iso-extraction proceeds is to first identify the cells intersected with the desired iso-surface, then extract the desired iso-surface cell by cell through outputting the approximated polygons. Therefore, there are two phases during the iso-surface extraction process, namely the cell searching and polygon (triangle) generation. While the second phase is quite standard now, except for the ambiguity problem, the first phase still leaves room for improvement.

There are essentially three schools of thoughts trying to reduce the cell searching time. The first type is called the space-based approach. Wilhelms et al. [WG92] proposed an octree decomposition method of this type for regular volume datasets. Each node in the octree records the min and max within it. A given iso-value is checked against the coarsest level in the octree and recursively sent to finer levels if necessary. However the real efficiency, usually defined by how much percentage of cells get touched , is very data-dependent and this method cannot be readily applied to irregular grids. The second type is called the range-based approach. Gallagher [Gal91] proposed a method of this type, which divided the range of data values into sub-ranges, cal led *buckets*. For each cell, we identify its starting bucket as well as its *span*, or the number of buckets its range intersects with. Cells with the same span are grouped together while within each group sub-groups are formed according to their starting cell. Span numbers greater than a threshold can be grouped together to save storage. Given a query, all the span groups are traversed and depends on the span that group represents, a number of more buckets will be traversed accordingly. Another very different method of this type, proposed by Livnat et al. [LSJ96], represents each cell by a point in a $2D$ plane with it

* Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan 106. Emails: {ckyang}@cs.ntust.edu.tw

y-coordinate to be the max value of a cell and the x-coordinate to be the min value of a cell. Each result to an iso-value query corresponds to a square region with its lower right corner touching the line defined by the equation of $x = y$. A *kd-tree* is used to build a hierarchy based on those points. At run time to answer a query the kd-tree is traversed to find the right and lower boundary and those points falling within this region are reported. Shen et al. [SHLJ96] later improved the searching time complexity further by using a uniform partition in the area of the $2D$ plane defined by $x < y$, the only area where all the cells of a data set can fall into. Yet another method, proposed by Cignoni et al. [CMM+97], demonstrated how to use the concept of *interval tree* to answer an iso-surface query, which is essentially a *stabbing query* in the field of *computational geometry* [Ede80], to achieve the optimal time complexity for searching. It built a hierarchical data structure so that an iso-surface query could be answered at run time by a logarithmic time complexity. The third type is called the surface-based approach, which at its preprocessing stage identifies a subset of the original dataset, called seed-set, and then at run time propagates to form the entire iso-surface from the intersected cells within the seed-set. The way of identifying a seed-set from the dataset distinguishes every method of this type. Itoh et al. proposed to build an a seed-set through an extrema graph, which is originally consisted of the local maximum and minimum points. These extremum points are connected to form a graph so that at run time, as each iso-surface must intersect with such a graph, the intersected cells can be located and propagated to form the entire iso-surface. However, as an iso-surface can be either closed or open, the above approach is suitable only the iso-surface is closed. In order to cope with the open iso-surface, boundary cells are sorted and included as well. [IK95]. The inclusion of boundary cells incurs great overhead. To address this issue, later they proposed a volume thinning approach to form a skeleton from the original dataset, and this skeleton serves as a seed-set [IYK96], thus eliminating the need of sorted boundary lists. By observing some basic property of a seed-set, Bajaj et al. begins with the whole dataset as a seed-set and gradually reduce the redundant cells with a sweeping paradigm [BPS96]. Their algorithm first defines the range of a face (edge or vertex) connecting two cells to be the iso-value range within which if one cell intersects with the corresponding iso-surface, the other cell will also be enumerated through the same face (edge or vertex) during the surface propagation process. Then for a cell, the fundamental property is that, if the union of ranges of its faces (edges and vertices) contains its range, the cell can be removed. Kreveld et al. developed an approximation algorithm [KOB+97] by constructing a *contour tree* which contains the local maximal, local minimal and saddle points, and for the first time, can be proved to generate a seed-set that is at most twice the size of the optimal seed-set size. However, the required running time is $O(NlogN)$. Our approach, on the contrary, may not be able to produce the seed-set as small as the contour tree approach does, but it is a linear time algorithm, and extremely easy to implement. Among the described related work, we will further detail the min-max span space and volume thinning approach in later sections as they serve as the foundations of our new approach.

3 Background

3.1 Volume Thinning

Itoh et al.'s volume thinning approach [IYK96] provides an efficient way to construct a seed-set. Its basic concept is to first identify the extremum points from a dataset. An extremum point is either a local maximum or minimum, in other words, a maximum or minimum compared to all its neighbors. Starting from the whole dataset, with those extremum points marked as non-eliminable, cells that will not affect local connectivity are removed. Here a cell denotes a cube which has eight scalar values defined on its eight grid points separately. The entire process proceeds as if the whole volume gets thinner and thinner, and at the end a skeleton is formed. See Figure 1 for a demonstration of a 2D thinning process.

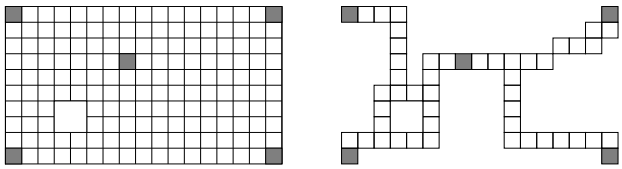

Figure 1: *A 2D volume thinning process. On the left: the original dataset, where the extremum points are marked in dark gray. On the right: The resulting skeleton after the thinning process.*

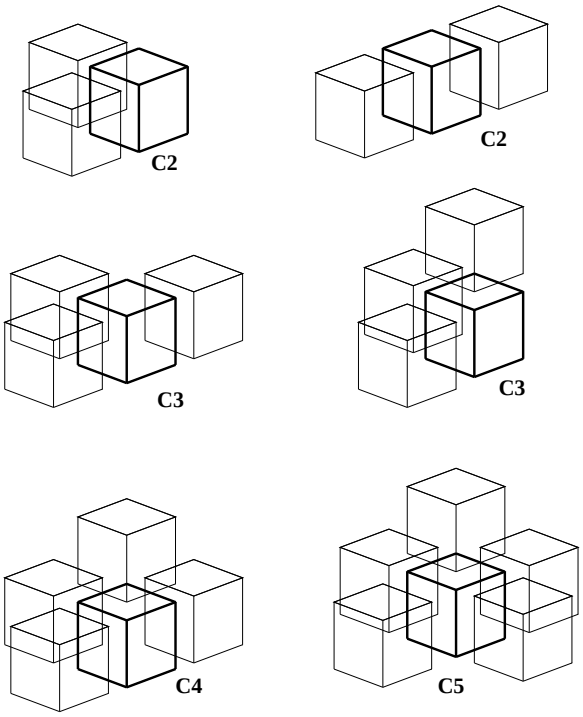

Figure 2: *Classification of a 3D volume thinning process, where Cn means the current cell has n face neighbors, i.e., n neighbors connecting to the current cell by one of its six faces. Notice that, according to the original volume thinning algorithm, unless a cell is belonging to class $C2$ or locating on the boundary of extremum points, it can be safely removed.*

From the implementation point of view, each cell is classified based on its face-connectivity with its neighbors. For instance, if a cell has two neighbors connected with it through one of its six faces, this cell belongs to the class of C2. Other classes could be defined similarly and accordingly. Based on the classification, the cell removing process could proceed as shown in Figure 2(re-drawn from the original paper). Notice that this approach always begins with those cells having the lowest face-connectivity, and in particular, cells belonging to class C1 can be removed in any cases. One thing we need to point out is that, in Figure 2, it is evident that only face-connectivity is considered. However, as an iso-surface could pass through two cells which are connected by an edge, edge-connectivity must also be taken into account, in the surface propagation phase. Similarly, an iso-surface could touch a vertex, or

a grid point, therefore vertex-connectivity should be considered as well. In fact, no matter what kind of connectivity is adopted, as long as it is used consistently with the surface propagation process, it is fine. In other words, if face-connectivity is used during the thinning process, then when propagating from the current cell towards other intersected cells, only those cells adjacent to the current cells through some faces should considered. Otherwise, inconsistency may be resulted.

There are some details worth mentioning. First, there may exist some *through holes* within the dataset, as shown by the empty region in Figure 1. Second, there may also exist some regions of *void*, i.e., a closed region whose interior contains all zero values. *Through holes* can be taken care of directly by the normal reduction process as shown in Figure 2, but a *void* area may consume indefinite amount of memory thus requiring a special "treatment". This operation is usually called a *pricking*, which randomly removes a cell from the composing surface around the void area. Third, when finding the extremum points, we may enter a homogeneous area where many extremum points are aggregated. To save memory resource, we could just keep one extremum point for each such contiguous region. This could be done by assigning each extremum point a unique ID and when enumerating a connected region by some traversal method (such as depth-first traversal or breadth-first traversal), only the extremum point with the smallest ID is retained, and other extremum points can all be discarded.

3.2 Min-Max Span Space

Min-Max span space was first proposed by Livnat et al. [LSJ96] to solve the cell searching problem in iso-surface extraction. As our new approach is inspired by this representation of a dataset, it is necessary that we explain its basic concept before illustrating our new idea. A min-max span space, as shown in Figure 3, is a 2D representation of a volume dataset, where each cell (cube or tetrahedron) is denoted by a point. The x coordinate of this point corresponds to the minimal scalar value defined on this cell, while y coordinate the maximal scalar value. As each cell's minimal scalar value cannot be greater than its maximal scalar value, all the points must occur within the region where the half plane corresponding to the equation of $x \leq y$. Moreover, using this representation, given an iso-value c, the cells intersecting with the desired iso-surface are those within the open regions defined by $x \leq c$ and $y \geq c$, as shown by the light-grey region in Figure 3.

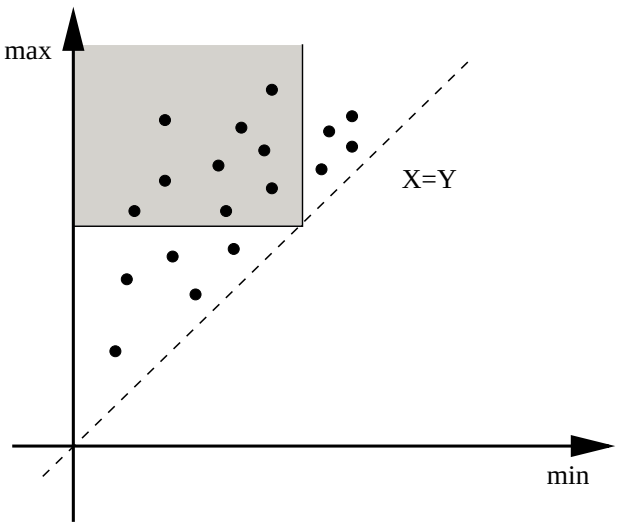

Figure 3: *A min-max span space of a dataset, where each cell is represented as a black dot. The grey area corresponds to an iso-value query, and the cells falling into this area intersect with the desired iso-surface.*

4 New Approach

In this section, we describe where our idea originates, and what our approaches are. We have implemented two variants, and each of them will be detailed in the subsections.

4.1 Upper-Left Envelope

Our new approach, though a surface-based algorithm, is in fact mainly inspired by the min-max span space representation of a data set. Recall that a seed-set of a dataset should bear the property that every iso-surface must intersect with it. As most surface-based approaches try to identify a seed-set from a dataset's original domain, what we are really curious about is how a seed-set behaves in the min-max span space domain. The first thing came to our mind is a line, which is parallel to the line of $x = y$. Apparently such a line satisfy the requirement: every iso-surface must intersect with it. However, given dots distributed on the upper-left half plane, it is not clear which line to choose and how a particular line corresponds to a seed-set. It did not take long before we realize that a perfect candidate for seed-set does exist, and it is in fact belonging to some forms of envelope line. In terms of min-max span space representation, we conclude that all those cells, represented as points, which have no cells on their upper-left side, must be included in the seed-set. Figure 4 demonstrates such an observation. In this figure, those points marked in circles must be included in the seed-set. This claim can be proved by the following. Assume S represents the set of all the points (cells) which does not have any other points on their upper and left-hand side, then this S must intersect with every iso-surface. Because as long as a iso-surface passes through a dataset, it must intersect with this data by at least one cell, say cell a. If cell a belongs to S, then we are done; otherwise there must exist another cell, say cell b, which is on the upper and left-hand side of cell a. From Figure 3 we know cell b must also intersect with this iso-surface. If cell b belongs to S, we are done; otherwise the procedure just described can be carried out recursively, and due to the finite cell number in a dataset, we will eventually reach a cell which belongs to S, thus proving our claim. For convenience, we will call these cells in S to be on the *upper-left envelope*. This is in fact a special case of the so called *maxima finding* problem [CHT03, BCL93]. Here we give a simple algorithm with worst-case time complexity of $O(NlogN)$, where N is the total number of cells in a dataset.

Observe that such cells must be located at the topmost positions for a given x coordinates in the min-max span space's representation provided there are more than one cells with the same x coordinates. We could first build two separate sorted lists based on the minimal and maximal value of each cell. Then we could traverse the sorted list of the minimal value, and when two cells are sharing the same minimal value, we retain only the one with bigger maximal value. The traversal itself takes $O(N)$ time while sorting taking $O(NlogN)$, thus the total time complexity of $O(NlogN)$.

Although it seems that we have found a perfect seed-set this way, there are still cells that should be included. In other words, the cells on the upper-left envelope are the cells that any seed-set must include, but the set of such cells is not yet complete to be a seed-set. Considering a 2D counterexample given in Figure 5. In this figure, the corresponding intervals for cell A, B and C are $(40, 50)$, $(50, 60)$, and $(30, 70)$. It is clear that only cell C is on the upper-left envelope while the other two are not. However, if we only retain cell C as the seed-set, then for the iso-value query whose iso-values falling in the interval of $(40, 50)$, there is no way of propagating

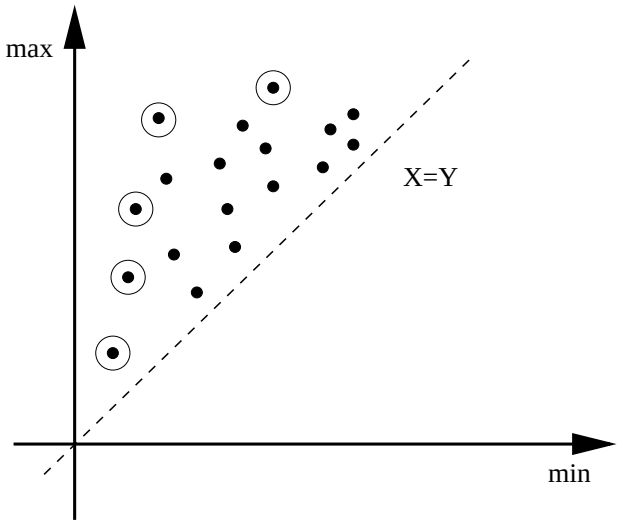

Figure 4: *A min-max span space of a dataset, where those dots who have no upper-left neighbors are marked with circles. These dots must be included in the final seed-set.*

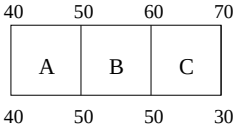

Figure 5: *A 2D counterexample. The numbers are the corresponding scalar values defined on the grid points.*

from cell C to cell A, as cell B does not intersect with such values. On a deeper thought, what is really missing here is the consideration of connectivity. Put it more concretely, cell B should be retained so that the iso-surface propagation can reach cell A.

4.2 Variant 1

To take connectivity into account, in variant 1 we make modification to our original algorithm as the following. We start by conceptually constructing a graph on top of the min-max span space where two cells (points) are connected by an edge if there are connected through a face. From this graph we try to remove unnecessary cells, as shown in Figure 6. In this Figure, a cell can be removed from the graph if it has an upper-left neighbor, because such a neighbor would have a smaller minimum and a larger maximum, therefore at run time, the cell can be *re-discovered* by the surface propagation process from this neighbor. Notice that once a cell is removed, the edges connected to it should be transitively adjusted, as shown in this Figure. From the implementation point of view, it is really not necessary to construct the connected graph, instead, we could start with treating the whole dataset as the seed-set, and then gradually remove unwanted cells one by one. Most importantly, *the algorithm requires only one pass of scan through all the cells then a seed-set can be constructed.* For each cell, we just need to check all its six face-connected neighbors to see if it has a *upper-left* neighbor in the min-max span space representation, or equivalently, if it has a neighbor whose range contains this cell's range. If so, the cell can be removed from the seed-set. Otherwise, it should be retained. The reason behind this is straightforward. If a cell has a face-connected neighbor which appears to its upper and left-hand side in the min-max span space, this means that neighbor has a range that includes the cell's range. This inclusiveness property

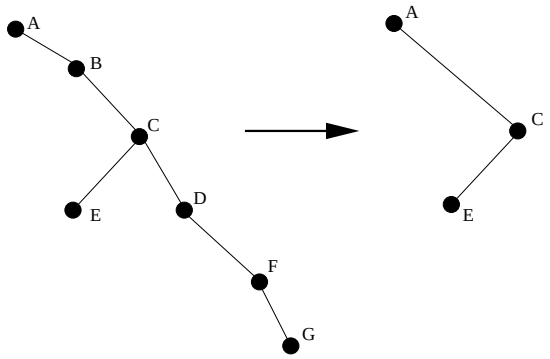

Figure 6: *A cell reduction process. On the left are connected components of the original dataset, while on the right are the reduced components of the dataset.*

guarantees that once that neighbor is preserved in the seed-set, the current cell could always be re-connected through the corresponding face. Notice the transitive property is implicitly preserved during this process, thus requiring no other bookkeeping or particular data structures. In other words, each cell could be checked individually without worrying its neighbors' existence. However, there is one exception. If cell A and cell B are adjacent with each other by a face, and if cell A and cell B have exactly the same range while all other neighbors of cell A and B do not have containing ranges, then our algorithm will remove cell B from cell A's point of view, and vice versa from cell B's point of view, as shown in Figure 7. One simple and less precise approach to deal with this exception is to first assign an unique ID to each cell, then when it comes to cell removal, only the cell with a bigger ID value is removed. However, this simple approach may produce a less optimal result, as shown in Figure 7, where both cell 1 and cell 5 will be retained, while only one of them should be preserved. To correct this, first we leave all such cells intact, then on each such connected region, as shown in Figure 7, we apply the cell propagation process as if we are to find all the intersected cells with an iso-surface. During this process, we could identify the cell with the smallest ID, and thus only such a cell should be retained, while all others could be safely discarded. Notice that the way we deal with the exception is exactly the same as we dealt with the problem of multiple connected extremum points. Although this incurs one extra pass of scanning, the affected portion is usually less than 50% of the whole dataset, see the performance evaluation section for detailed results.

There is one more optimization that we could perform to further reduce the size of the seed-set. Recall in the iso-surface propagation process, the intersected cells found in the seed-set are used to propagate to locate all the intersected cells with the desired iso-surface. Usually this propagation is performed through face connectivity, however, as an iso-surface could touch a vertex or pass through an edge, we could modify the surface propagation process accordingly. This modification also affects the seed-set construction as the definition of a *neighbor* of a cell gets changed. By taking the new definition into account, our algorithm requires little modification while most of it remains unchanged.

Note that there is one more modification to be done during the surface propagation process so that our approach is feasible. Refer to Figure 8 for a 2D illustration. In this Figure, the curves represent the iso-surfaces corresponding to iso-value equal to 25. According to normal surface traversal, such as the one in Bajaj et al. [BPS96], each cell only checks to see if any of its faces (edges or vertices) intersects with the iso-value, surface propagation will proceed along that face (edge or vertex) neighbor. Without modification, it is apparent that a surface propagation starting from cell C will not reach

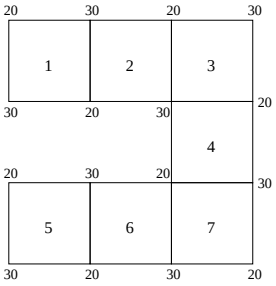

Figure 7: *A 2D example where several cells with exactly the same range are connected together.*

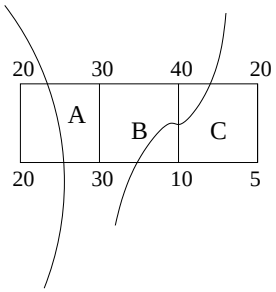

Figure 8: *A 2D example of iso-surface(s). Here the given iso-value is* 25.

cell A, as the face between cell A and B does not intersect with the given iso-value. To correct this, we just need to make little modification to the surface propagation process: if the range of any face (edge or vertex) neighbors of the current cell intersects with the given iso-value, surface propagation should proceed along that neighbor.

4.3 Variant 2

Recall the method proposed by Bajaj et al. [BPS96], where a cell could be eliminated if the union of ranges of its faces (edges or vertices) contains its range, then this cell could be eliminated. By making use of this property, Bajaj et al. applied a sweeping algorithm to obtain an approximate seed-set. Similar to the modification we just made during the surface propagation process, we could modify Bajaj et al.'s cell removal property to the following. If the union of ranges of a cell's neighbors contain its range, then this cell can be removed. Through the representation of min-max span space, we found that this union property could be checked individually thus greatly simplifying the checking process, that is, there is really no order of dependence. No matter from which cell do we start the checking, we end up with the same resulting seed-set. Furthermore, the checking could be easily merged into our variant 1 of algorithm, thus readily forming the variant 2 algorithm.

The reason that the union property could be checked individually is the following. From the representation of min-max span space, if the union of ranges of a cell's neighbors contain the cell's range, this means *this cell must have a upper-left neighbor (just like in variant 1), or the union of ranges of its upper and left neighbors must contain the cell's range*. As this relation between two cells is *antisymmetric*, and when two cells are having exactly the same ranges will be dealt with (by surface propagation) separately at the end of the elimination process, the property check can be performed individually on every cell. More concretely, whenever we find a cell whose range can be covered by its neighbors, we do not need to worry if any of neighbors exists or not at the time of processing a cell; because if any of its neighbors really gets removed, that neighbor itself must have its own covering neighbors. For implementation, we adopt the following algorithm, which could accommodate the variant 1 algorithm at the same time. For each cell, we calculate the *minimum of minimum* of all its neighbors appearing to its top, denoted by MIN_OF_MIN, and the *maximum of maximum* of all its neighbors appearing to its left, denoted by MAX_OF_MAX. If $MIN\_OF\_MIN \leq MAX\_OF\_MAX$, it can be shown that the union of its neighbors' range contains its range. Note that the case of having upper-left neighbors is already implicitly included.

4.4 Combined with Volume Thinning

In addition to proposing a new approach, another goal of this paper is to combine this approach with existing approach to seek for an even smaller seed-set. According to Itoh et al. [IYK96], their volume thinning approach performs much better than their previous extrema graph approach [IK95], due to the elimination of boundary cells. Therefore, we re-implement the volume thinning algorithm so that we could compare and combine their approach with ours. There is one immediate optimization that we could perform and sometimes it can significantly reduce the skeleton size produced by the original thinning algorithm (as will be demonstrated in the performance evaluation section). The trick is when the eight scalar values of a non-isolated cell are all equal to one constant value, this cell can be removed, because this value must also appear on on of its neighbors. At run time, when this particular iso-value is requested, one of its neighbors should be included, either because that neighbor belongs to the seed-set, or is reached by the propagation from a seed-set, and thus this cell will eventually be included as well. Therefore this cell does not need to be present in the seed-set.

Noticing that volume thinning and our approach bear different perspectives towards cell reduction, we combine these two approaches and see how the merged variant performs. What we could do is to first generate a skeleton by using the volume thinning approach, then apply the procedure that we mentioned in the last section, i.e., each cell in the skeleton checks to see if it has a face-connected, edge-connected, or vertex-connected neighbor whose range containing its range. And if so, the cell is removed; otherwise it is retained. Other implementation details such as equal-valued adjacent cells can be handled in exactly the same way.

As the last remark, after the seed-set is constructed, we could build an *interval tree* from the cells in the seed-set to further speed up the seed cells searching at the run time, just as proposed by [BPS96]. Because a seed-set is usually relatively smaller compared to the original data set, the cell searching time for finding the intersected cells in the seed-set could become extremely fast. Notice that although these intersected cells still need to be propagated to find all the cells intersected with a given iso-surface, nevertheless, the asymptotic time complexity is roughly only proportional to the number of cells intersected with the iso-surface. That is, we only spent minimal effort on those non-intersected cells, and at the same time could enumerate cells which facilitate the generation of triangle strips to enhance the rendering throughput of the graphics engine.

5 Performance Evaluation

We have implemented our system on a Pentium 4 2.8GHz machine with 1GByte memory, running the Windows 2000 Professional operating system. We have collected and tested totally 10 volume datasets. Table 1 lists the characteristics of these 10 datasets. We also list the number of extremum points for reference. # of Skeleton Cells are the results by using Itoh et al.'s volume thinning algorithm, which we have re-implemented for comparison study. For convenience, we use *Whole n* to denote the result of using variant

Data set	# of Skeleton Cells	# of Skeleton Cells After Removing constant cells
MR Brain	2160246	2109589
CT Head	1410784	1410592
CT Engine	791428	776573
SOD	41902	27611
HIPIP	10520	10518
Hydrogenatom	1732	943
Aneurism	33134	12726
Bonsai	155850	115281
Skull	495554	491410
Foot	234405	200949

Table 2: *Comparison of the number of seed cells between the original skeleton and the new skeleton after removing the constant cells.*

n algorithm running on the whole dataset, and similarly *Skeleton n* the result of using variant n algorithm running on resulting skeleton produced by the volume thinning approach.

Table 2 shows the impact of removing those cells with constant values. As shown by the *Aneurism* dataset, the number of skeleton cells could become roughly three times smaller. This phenomenon is not completely occidental, as shown by Table 1, only 256 possible values to be distributed to $127 \times 127 \times 127$ cells, there may still exist some homogeneous regions which were not removed by the volume thinning process.

Table 3 demonstrates how connectivity affects the seed-set size, by using the variant 1 algorithm running on the whole dataset. In this paper, we assume each cell is a cube, therefore it has 6 faces, 12 edges and 8 vertices. If we allow connectivity to be extended from only face-connectivity, denoted by (F), to face-connectivity plus edge-connectivity, denoted by $(F + E)$, or together with vertex-connectivity, denoted by $(F + E + V)$, for both the seed-set reduction and surface propagation, then the seed-set size could be reduced up to four times smaller, as the *Hydrogenatom* dataset shows in this table. As this table suggests, from this point on all the reported numbers are based on the $(F + E + V)$ connectivity.

Table 4 juxtaposes the resulting seed-set sizes by different algorithms, where *Itoh* and *Bajaj* represent the results by the work in [IYK96] (volume thinning) and [BPS96] (fast iso-contouring), separately. Due to the limit of time, we did not get to re-implement Bajaj et al.'s method [BPS96], so we extract the number from the paper and downloaded all the datasets available for our comparison. There are several relationships to observe from this table. First, the numbers in *Whole 1* and *Skeleton 1* are all less than the numbers in *Whole 2* and *Skeleton 2*. This is simply because in the variant 2 algorithm we look for more chances to delete a cell, and the test in variant 1 algorithm is just a special case of that in variant 2 algorithm. Second, the numbers in *Skeleton 1* and *Skeleton 2* are smaller than those in *Itoh* as the former two are built on top of the latter one to seek further possibility of reduction. The most intriguing part is the comparison between *Whole 2* and *Skeleton 2*, where *Whole 2* wins most of the time while only loosing for the *HIPIP* and *Hydrogenatom*. And the same time, these two datasets are also where *Whole1* lost most when compared to *Itoh*. After a detailed analysis we found because these datasets present thin layers of equal values which may be pricked into fragmented parts by the *Whole 2* approach, but can be handled properly by the volume thinning approach.

For the comparison between *Whole 2* and *Bajaj*, the reason why *Whole 2* wins all the time is also self-evident, as explained previously that the tests performed in *Bajaj* can be deemed as a special case of what is performed in *Whole 2*. Overall, the *Whole 2* can reduce the dataset size by up to 15 times smaller (as shown by the *Aneurism* dataset) when compared with *Itoh*, and up to 3 times smaller (as shown by the *MR Brain* dataset) when compared with *Bajaj*.

Data set	Time of Whole 2	Time of Skeleton 2
MR Brain	6.047	2.750
CT Head	5.625	2.765
CT Engine	5.500	3.234
SOD	0.859	0.781
HIPIP	0.172	0.063
Hydrogenatom	1.109	0.922
Aneurism	0.641	0.546
Bonsai	1.109	0.813
Skull	1.828	1.281
Foot	1.140	0.844

Table 5: *Running time of the variant 2 on all datasets, with only cell searching time reported. Numbers are in seconds.*

Data set	% of Equal-Valued Cells in Whole 2	% of Equal-Valued Cells in Skeleton 2
MR Brain	12.4	10.7
CT Head	19.9	11.5
CT Engine	21	35.5
SOD	15.8	16.5
HIPIP	7	8.2
Hydrogenatom	73	45.8
Aneurism	90.1	22.3
Bonsai	53.3	23.8
Skull	2	16.4
Foot	55.6	15.1

Table 6: *Percentage of dealing with equal-valued cells under variant 2 of our approaches running on the whole dataset and the skeleton.*

Table 5 presents the timing results of our variant 2 algorithm, so far the best implementation of ours. It makes use of the $(F + E + V)$ connectivity, removes the cells with constant values and equal-valued cells, and performs the union property test. We do not report the triangle interpolation time, as the focus of this paper is on the seed-set generation. We also do not include the skeleton generation time by the volume thinning approach as it is not our contribution. In fact, our code is still far from being optimized. Nevertheless, these results show that with a moderate class of PC, all the seed-sets could be generated with a reasonable speed. And most important of all, these constructed seed-sets could be stored or even used for building an *interval tree* to quickly answer repeatedly iso-value queries.

Table 6 reports the percentage of dealing with equal-valued cells. This incurs a slight overhead on processing those equal-valued cell regions, but the overall time complexity is still linear in terms of the input dataset size. As remarked before, results shows that usually the percentage is below 50%. The reason we have over 50% of equal-valued cells is really data-dependent. The *Hydrogenatom*, *Aneurism*, *Bosai* and *Foot* datasets are those with a significant portion of zero-valued area, thus the big percentage of equal-valued cells. That is also why when running on top of the skeleton produced by the volume thinning approach, the percentages usually drop significantly. On the other hand, higher percentage also means a higher chance that those equal-valued cells being removed, as can be seen from Table 4.

To prove correctness, we have also verified the resulting seed-sets produced by all variants of our algorithms. We have devised a

Data set	Dimension	Range	# of Extremum Points	# of Skeleton Cells
MR Brain	256 × 256 × 109	0 ∼ 65535	1489073	2160246
CT Head	256 × 256 × 113	0 ∼ 65535	764819	1410784
CT Engine	256 × 256 × 110	0 ∼ 255	220783	791428
SOD	97 × 97 × 116	0 ∼ 255	2758	41902
HIPIP	64 × 64 × 64	-0.55625 ∼ 0.58136	1465	10520
Hydrogenatom	128 × 128 × 128	0 ∼ 255	44	1372
Aneurism	128 × 128 × 128	0 ∼ 255	6034	33134
Bonsai	128 ×128 × 128	0 ∼ 255	43479	155850
Skull	128 ×128× 128	0 ∼ 255	276530	495554
Foot	128 ×128 × 128	0 ∼ 255	127554	234405

Table 1: *Characteristics of input data sets used in this performance study.*

Data set	Whole 1(F)	Whole 1 (F+E)	Whole 1 (F+E+V)
MR Brain	656908	335590	283335
CT Head	832534	428352	356679
CT Engine	838520	487104	419687
SOD	154338	80523	62619
HIPIP	92582	53486	41262
Hydrogenatom	112412	41122	27891
Aneurism	5411	3589	3241
Bonsai	63234	32626	27931
Skull	243459	119357	98049
Foot	73504	39055	32878

Table 3: *Comparison of the number of seed cells among the variant 1 of our approach running on the whole dataset, under different connectivity implementations.*

Data set	Itoh	Whole 1	Whole 2	Skeleton 1	Skeleton 2	Bajaj
MR Brain	2160246	283335	209332	425159	307039	639891
CT Head	1410784	356679	162069	334120	281006	423366
CT Engine	791428	419687	115918	284245	220231	180048
SOD	41902	62619	7530	17807	11294	13004
HIPIP	10520	41262	3930	5671	3652	4616
Hydrogenatom	1732	27891	3141	763	400	N/A
Aneurism	33134	3241	2507	5973	4061	N/A
Bonsai	155850	27931	18970	41039	29942	N/A
Skull	495554	98049	47983	120530	93304	N/A
Foot	234405	32878	24617	50267	36708	N/A

Table 4: *Comparison of the number of seed cells between Itoh et al.'s volume thinning approach and variant 1 of our approach running on the skeleton produced by Itoh et al.'s approach, under the connectivity of face, edge and vertex.*

way to check if seed-set is indeed a seed-set by testing all possible iso-values. For datasets with only integer scalar values, we just need to test each integer within the scalar value range. For datasets with floating point scalar values, we first find the union of all floating point scalar values, then exhaustively perform iso-value query with values coming only from the set of union. It can be shown that by testing such values, we could enumerate all possible cases of how all the cells of a dataset intersect with all possible iso-surfaces.

To summarize, we have proposed an algorithm which makes the following contribution. First, it is extremely simple to implement, while at the same time performs much better than most existing algorithm. 2. It generalizes some existing scheme (such as the one by Bajaj et al. [BPS96]) while still preserving the linear time complexity for building a seed-set at the preprocessing time. 3. It can be easily combined with other approaches, such as volume thinning, to further reduce the size of a seed-set. 4. It adds two minor optimizations by removing the cells of constant values and equal-valued cells, which sometimes may help reduce the seed-set significantly.

6 Conclusion

We have proposed and implemented a new approach to identify a seed-set from a volume dataset. This approach, though very simple, takes just linear time of preprocessing to construct a relatively small seed-set. Most importantly, due to its simplicity, it could also be combined with other seed-set finding approaches. In particular, our approach could be applied together with the volume thinning approach, to yield an even better result than applying volume thinning alone. Overall our algorithm could reduce the seed-set size by up to 15 times smaller when compared with the original volume thinning approach. There are two directions that we plan to pursue in the future. First, we will generalize our work to handle tetrahedral volume datasets as well. Although tetrahedral volume datasets present more complex topology, in terms of face, edge and vertex connectivity, it is in fact simpler than the case of regular volume datasets. Second, just like the work done by Kreveld et al. [KOB$^+$97], we will work on deriving an approximation algorithm which could find a seed-set with provably small size; however, we wish to lower the time complexity from $O(NlogN)$ to $O(N)$.

References

[BCL93] J. L. Bentley, K. L. Clarkson, and D. H. Levine. Fast Linear Expected-Time Algorithms for Computing Maxima and Convex Hulls. *Algorithmica*, 9:168–183, 1993.

[BPS96] C. L. Bajaj, V. Pascucci, and D. R. Schikore. Fast Isocontouring for Improved Interactivity. In *Proceedings on 1996 Symposium on Volume Visualization*, pages 39–46, October 1996.

[BpS97a] C. L. Bajaj, V. pascucci, and D. R. Schikore. Fast Isocontouring for Structured and Unstructured Meshes in Any Dimension. In *IEEE Visualization '97 Late Breaking Hot Topics*, 1997.

[BpS97b] C. L. Bajaj, V. pascucci, and D. R. Schikore. Seed Sets and Search Structures for Accelerated Isocontouring. Technical Report 97-034, Department of Computer Science, Purdue University, 1997.

[CHT03] W. Chen, H. Hwang, and T. Tsai. Efficient Maximal-Finding Algorithms for Random Planar Samples. *Discrete Mathematics and Theoretical Computer Science*, 6:107–122, 2003.

[CMM$^+$97] P. Cignoni, P. Marino, C. Montani, E. Puppo, and R. Scopigno. Speeding Up Isosurface Extraction Using Interval Trees. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):158–170, April 1997.

[Ede80] H. Edelsbrunner. Dynamic Data Structures for Orthogonal Intersection Queries. Technical Report Report F59, Inst. Informationsverarb.,Tech. University Graz, 1980.

[Gal91] R. S. Gallagher. Span Filtering: An Optimization Scheme for Volume Visualization of Large Finite Element Models. In *IEEE Visualization '91*, pages 68–75, October 1991.

[IK95] T. Itoh and K. Koyamada. Automatic Isosurface Propagation Using an Extrema Graph And Sorted Boundary Cell Lists. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):319–327, December 1995.

[IYK96] T. Itoh, Y. Yamaguchi, and K. Koyamada. Volume Thinning for Automatic Isosurface Propagation. In *IEEE Visualization '96*, pages 303–310, October 1996.

[KOB$^+$97] M. J. Kreveld, R. Oostrum, C. J. Bajaj, V. Pascucci, and D. Schikore. Contour Trees and Small Seed Sets for Isosurface Traversal. In *Symposium on Computational Geometry*, pages 212–220, 1997.

[LC87] W. E. Lorensen and H. E. Cline. Marching Cube: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[Lev88] M. Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, 8(3):29–37, March 1988.

[Lev90] M. Levoy. Efficient Ray Tracing of Volume Data. *ACM Transactions on Graphics*, 9(3):245–261, 1990.

[LSJ96] Y. Livnat, H. Shen, and C. R. Johnson. A Near Optimal Isosurface Extraction Algorithm Using the Span Space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, March 1996.

[SHLJ96] H. Shen, C. D. Hansen, Y. Livnat, and C. R. Johnson. Isosurfacing in Span Space with Utmost Efficiency. In *IEEE Visualization '96*, pages 287–294, October 1996.

[WG92] J. Wilhelms and A. V. Gelder. Octrees for Faster Isosurface Generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.