Exploded View Diagrams of 3D Grids

Zamir Martins Filho
University of Calgary
Calgary, Canada
Email: zamirmf@gmail.com

Emilio Vital Brazil
University of Calgary
Calgary, Canada
Email: evbrazil@ucalgary.ca

Mario Costa Sousa
University of Calgary
Calgary, Canada
Email: smcosta@ucalgary.ca

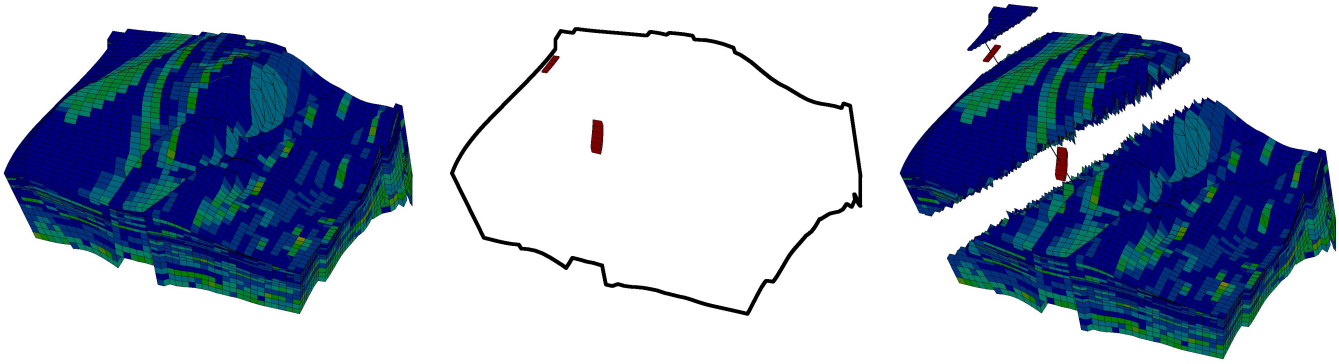

Fig. 1. Exploded View Diagrams applied to a Corner-Point grid which represents a geological model (left to right): the whole model; the objects of interest; the final result.

***Abstract*—We present a system for creating interactive exploded view diagrams in generalized 3D grids. The primary difference between our approach and existing ones is that our technique neither requires geometrical information of the whole model nor any information regarding the relationship among model parts; instead our implementation depends on which grid cells are considered as object of interest, and which view angle to use. To achieve this, we introduce the *Explosion Tree*, a data structure closely related to a BSP tree, which supports the explosion view diagrams technique based on the relationship between disjoint convex polygons. In this paper we discuss the application of this technique to *Corner-Point Grid* which has been extensively used for geological modeling and flow simulation. All the data presented in this work consists of real data currently used in the industry.**

***Keywords*-Computational Geometry; Computer Graphics;**

I. Introduction

3D grids have been widely used to represent data in various domains [1]. The oil & gas domain massively uses 3D grids and among their different type of grids the *Corner-Point Grid* is one of the most challenge grid types (see Figure 3). In the realm of reservoir engineering, those grids are used to model geological features, therefore, domain experts are constantly looking for better ways of visualizing those grids. Through an insightful visualization, domain experts can explore, identify, and analyze information which might be crucial in decision making, in a domain where mistakes on the interpretations can cost millions of dollars. One of the challenges of visualizing 3D grids, and more particularly irregular grids is that their visualizations are often faced with problems caused by occlusion in 3D space. The fact that they are irregular and often have discontinuities implies that general techniques for volumetric data cannot be applied (due to the fact that it assumes the regularity of the data). In addition to that, grid cells represent properties values (e.g., porosity, permeability) through color mapping, therefore transparency-based techniques make difficult to correlate grid cells with a respective color map or legend (see Figure 2).

A. Corner-Point Grid

3D grids have been extensively used by various domains, turbulence modeling, wind and water tunnel modeling, flow simulation and geological modeling are some examples [1]. Moreover, the last two domains are usually based on the corner-point grid [2]. In comparison to regular grids, corner-point grids reduce the numerical error by adjusting themselves closer to the actual geometry, therefore, geological features are modeled through a corner-point grid because of the benefits from the engineering point of view, not from a computer science perspective. Corner-point grid has been defined as standard by the flow simulation industry, thus there is an extensive amount of data to be analyzed [3]. Corner-point grid can be classified as a distorted grid and also as a flexible grid. As a non-regular grid it can represent irregular geometries in a more precise way than a regular grid. Although it contains eight vertices (see Figure 3 right), no spatial assumption can be made regarding them, two or more vertices can have the same location and they can also contain discontinuities [2](see Figure 3 left). While in regular grids, it is feasible to predict spatial location based on the cell size and the amount of cells,

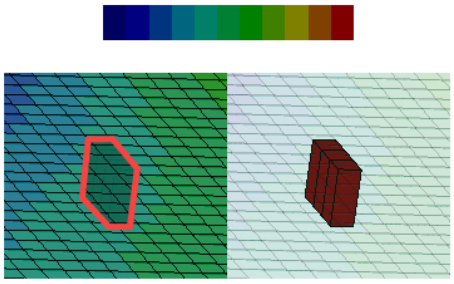

Fig. 2. Transparency makes difficult to correlate grid cell with the legend: (top) the legend and its actual colors; (bottom-left) occluded grid cells colors are unclear while it is possible to correlate the remaining cells colors with the legend; (bottom-right) in order to make occluded grid cells colors easy to be correlated, the remaining cells colors become unclear.

it is not possible with the corner-point grid. An introduction of this grid from a computer graphic point of view can be found at [4]. It is important to remark that apart of its qualities, nowadays corner-point grids have a key role in the industry due by the huge number of existing models.

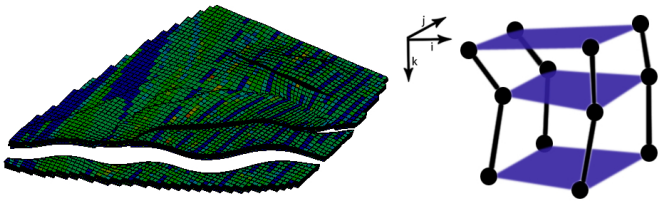

Fig. 3. The corner-point grid (from left to right): An instance of a corner-point grid which includes discontinuities, representing a geological model and its faults; a diagram of corner-point grid cells depicting the irregularity among the grid cells caused by the uneven distribution of vertices that represent their corners.

One approach toward the analysis is by selecting cells with similar behavior, i.e., cells with properties within a specific range of values [4]. It is useful to identify outliers and how an specific property is behaving among a grid cell and its neighbors. Therefore, by selecting a range of values, a domain expert wants to focus on the grid cells which fall into this range. Those cells are considered as primary objects, while the remaining as secondary objects (see Figure 4). The number of primary objects can go from zero to the total number of cells in the grid. Since the criteria of selection can be changed, the number of primary objects can change at anytime.

Although the domain expert has decided to focus on primary objects, it does not mean that secondary objects can be discarded from the visualization. For the expert, it is still important to be able to explore the surroundings of those primary objects from different angles. Each grid cell clipped by the visualization technique implies that some part of the information is also lost. This aspect plays an important role into the deciding of which visualization technique should be used.

B. Explosion Diagrams

One of the main challenges of 3D objects is occlusion. Most proposed solutions to this problem classify objects of interest as primary and the remaining as secondary, keeping all the primary object data while compromising on some amount of data related to the secondary objects. Depending on the type of application, partial visualization is acceptable and for those cases, illustrative visualization techniques such as Ghost-view [5] and Cut-away view [6], [7] can be applied. An application of cut-away view technique in corner-point grids can be found at [4]. On the other hand, in the case of applications where partial data visualization is not acceptable other techniques need to be explored and one such technique is exploded view diagrams. In this technique secondary objects are displaced and even split, however all data and its overall correlation are kept. Although much has been done toward volumetric data and CAD models, there is a lack of work applied to non-regular grids.

Originally, exploded view diagrams have been applied in models where all information regarding their geometry is well known. Most previous approaches perform this technique by using an additional information regarding how model's parts are related to each other (part hierarchies). Also it is important to note that in the previous applications parts were static, meaning that their geometry never change, hence the relationship between parts was also static. Exploded view diagrams [8] have been proposed to solve the occlusion problem in CAD models and volumetric data.

In Li et al. [8], object parts and its hierarchy are used as input data, then based on how parts are related to each other, the direction and displacement magnitude are calculated to achieve the final effect. They assumed that camera positions are limited and those are pre-computed to speed up the interaction with the data by reducing the amount of calculation. The data structure used is a complementary to the one proposed by Agrawala et al. [9]. The main difference in our approach is that we do not require object parts, part hierarchy, information related to secondary objects nor pre-computation to produce the same effect. Some other work related to exploded view diagrams are: Li et al. [10] that presents a solution to 2D exploded view for images; Bruckner and Groller [11] applied physical equations to add extra effect to the exploded view and in most cases they split the secondary object into fixed number of pieces; Karpenko et al. [12] explores exploded view diagrams applied to parametric surfaces and Tatzgern et al. [13] extends [8] in the sense of choosing which parts to be exploded. Different from the exploded view, some approaches try to tackle the problem of occlusion by applying a spherical force field; Elmqvist [14] presents a prototype which is suitable when unrelated 3D objects share the same scene and user can use a 3D pointer to explore the space. McGuffin et al. [15] use the concept of semantic layers to create the concept of parts in volumetric data.

While just a few of them would be applicable to regular grids (as volumetric data can be seen as), none of them takes in

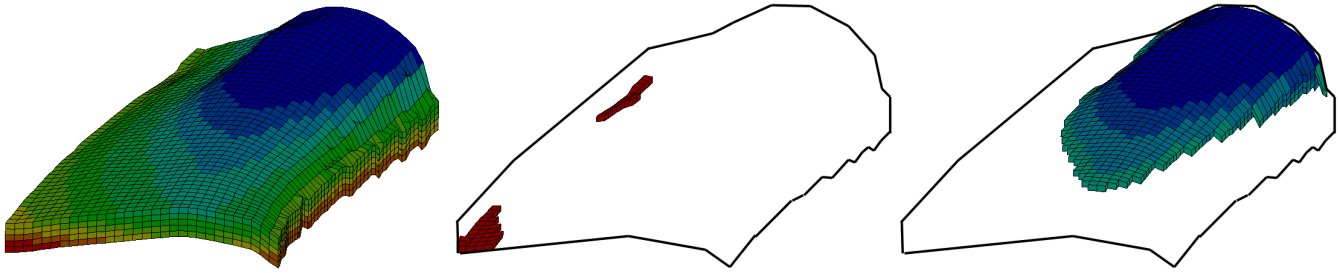

Fig. 4. Selection based on property (i.e., pressure) values (from left to right): The whole model ranging from 186 kPa to 1317kPa; When the range is between 13100 and 13137 kPa the amount of primary object is reduced; Changing the range to values between 186 and 1023 kPa, also change how many and which cells are defined as primary objects.

consideration the possible irregularity and discontinuities of an irregular grid. Also none of them consider that the relationship between primary and secondary objects can be changed.

C. Contributions

In this work we propose a technique to show occluded objects inside 3D grids (including non-regular grids) based on illustrative exploded view diagrams [16].We propose a solution which only information regarding the primary objects and gaze are required, supporting the exploded view technique in a dynamic environment, it means that the relationship between primary and secondary objects can change anytime. It is important to observe that we do not use information regarding secondary objects as the previous approaches for computing explosion diagrams. Thus we believe that in the realm of explosion diagram techniques, our approach is different.

As a core of our approach, we have created *Explosion Tree*, a data structure which receives a set of disjoint convex polygons (derived from primary objects) as input and returns a final position to all vertices (including those that belong to the secondary object) as an output. The data structure is the core of our application and it contributes to the major goal of creating a final effect similar to technical illustration based on the traditional exploded views technique [16], having limited amount of information in a dynamic environment. Our data structure (i.e., *Explosion Tree*) is an extension of a binary space partitioning tree (i.e., BSP-tree) that is basically a binary tree of planes which subdivide the space [17]. Instead of considering all nodes similar, we extend this concept by categorizing nodes into two different classes and by adding a key operation which traverse the tree setting the offset values for all node, where those offsets are responsible to translate each vertex in order to find its final position avoiding overlapping, and exposing all primary objects.

This work has two contributions. The first is a new data structure (i.e., *Explosion Tree*) which calculates final vertex positions for exploded views diagrams supporting conventions from traditional illustration. This data structure requires less information than any of the previous approaches. The second is the application of the exploded diagrams technique to 3D grids where there is limited amount of information, also not supported by previous approaches.

The rest of the paper is organized as follows: section II presents a system which wraps our data structure and propose an effect similar to the Exploded Diagrams applied to 3D grids; finally we present the results and conclusion on sections III and IV respectively.

II. System Architecture

We developed a system that on the same time that it wraps our data structure, *Explosion Tree*, it applies the exploded diagram effect to corner-point grids. The system supports the grid geometry in a dynamic scenario, i.e., the user can select different ranges of property values so then changing the primary objects. Using the range we classify objects as primary and secondary, our system use the primary object geometry to calculate the final position (exploded diagram) of the whole model. Animation between the initial and final positions and a context line are provided in order to improve the visual correlation between primary and secondary objects. In addition, the user can lock the effect and navigate in the 3D scene.

A. Primary and Secondary Objects Definition

Grid cells can be selected as primary objects based on different constraints. In this work we focus on the same criteria that is used in the oil & gas domain, i.e., as each grid cell is associate to at least one property value (i.e., pressure, porosity, permeability), the selection is performed by specifying a range of values of a property and cells which values fall in this range are selected as primary objects. The remaining cells or cells properties with values outside of the selected range are considered as secondary objects (see Figure 4). After the selection is performed all grid cells are going to fall under one of the two previous mentioned categories. It is important to note that the only grid cells that are going to be taking into further consideration are those classified as primary objects.

B. Convex Polygons Definition

After having all grid cells classified, a 3D clustering algorithm is applied only to those cells set as primary objects. The goal of the algorithm is to define which primary objects compose each cluster. More specifically, it runs through all primary objects and those which share at least one vertex are defined into the same cluster. Having clusters defined, the next

step is to convert 3D clusters into a set of 2D convex polygons, and it is achieved by projecting (orthographic projection) each cluster to view plane following by storing each of them into one buffer. For each buffer a pixel-based convex hull is created and then, an intersection verification is performed between all convex polygons. In cases where there is an intersection between two or more convex polygons a new convex polygon is calculated based on all intersecting polygons and from now on it will be used replacing those intersecting convex polygons (see Figure 5). The meaning of merging intersecting convex polygons is to reduce the total number of objects passed to our data structure and to isolate the processing of each view angle from the general processing. Therefore, while 3D clusters remain unchanged, the convex polygons arrangement is going to depend mainly on the view angle. Changes in the view angle require the projection and intersection test steps to be performed.

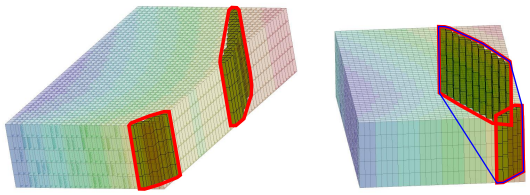

Fig. 5. Convex polygons (originated from 3D clusters): (left) two non-intersecting convex polygons, red lines; (right) the same 3D clusters projected from a different gaze originate two intersecting convex polygons (red lines), which are going to be replaced as one (blue line).

C. Initializing the Explosion Tree

As it is going to be described in this section, a set of planes is created based on the relation between convex polygons and the order of comparison implies in the arrangement of planes that happens inside the data structure. In this work, convex polygons are sorted based on the lexicographic sorting in (x,y), i.e., first it takes into consideration the x-axis and then the y-axis (if a draw on the x-axis occurs). Therefore, the last step before sending data as input to the *Explosion Tree* is to sort the disjoint convex polygons.

So far, as secondary objects are not taken into consideration, performance is influenced by the number of 3D clusters and the total amount of primary objects among them. The sorted disjoint convex polygon set is the input data to our data structure and the first step performed by this data structure is to find a set of 2D planes minimizing the number of convex polygons per region. Similar to a BSP-tree, each plane is going to define two regions (one at each side), then regions are subdivided until there one convex polygon per region or until there is no more valid planes (our stopping criteria). A valid plane is a plane which divide a region in two non-empty regions and does not intersect any convex polygon. In order to keep an aesthetic aligned to the exploded view technique, we define a plane in the point x_0 between two convex polygons 1 and 2, in a way that the point x_0 is the middle point of the line segment $l - \{\Omega_1 \cup \Omega_2\}$ (see Figure 6). Also, we define the first normal n as a vector between the convex polygon centroids (i.e., $\{c_{2_x} - c_{1_x}, c_{2_y} - c_{1_y}\}$), and while there is no valid plane we keep rotating the normal in a thirty degree increment (see Figure 7). Note that if there is no valid plane between two or more convex polygons they are going to share the same region. Afterwards, more planes are added to the tree until one of our stopping criteria is reached (see Figure 8). The data structure now stores which convex objects belongs to each region and also the sequence of nodes to reach it from the root downward.

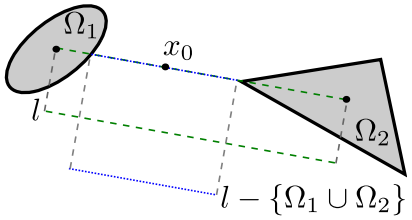

Fig. 6. Point x_0 is defined as the point in the middle of the line segment $l - \{\Omega_1 \cup \Omega_2\}$.

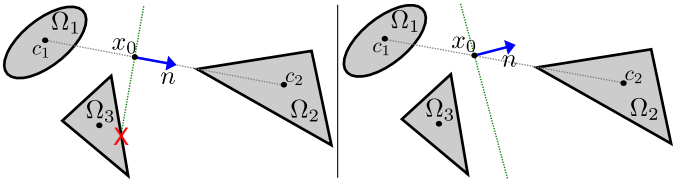

Fig. 7. Plane definition:(left) the first normal n is defined as $c_2 - c_1$, however when it intersects polygons belonging to the same region it is rotated by thirty degrees and a new test is performed; (right) this plane does not intersect any polygon therefore, there is a valid plane between convex polygon Ω_1 and Ω_2.

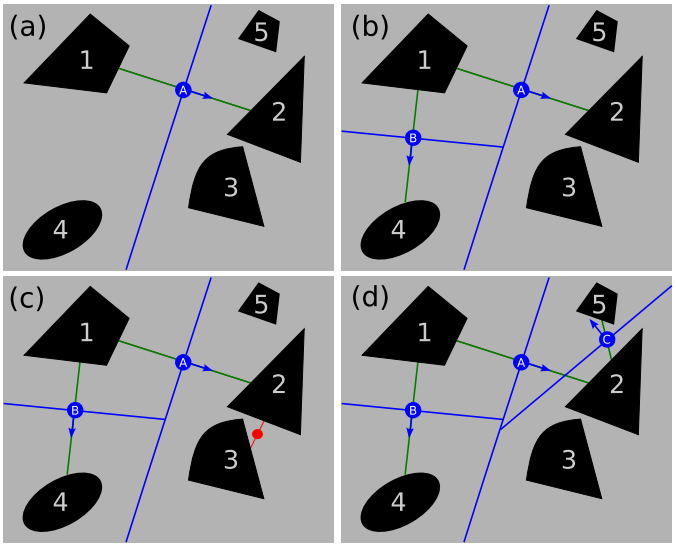

Fig. 8. Build sequence (a) the first plane, A, is defined between convex polygons 1 and 2; (b) on the left side of plane A, a plane B is defined between 1 and 4; (c) on the right side of A no plane is found between 2 and 3; (d) still on the right side of A plane C is defined between 2 and 5.

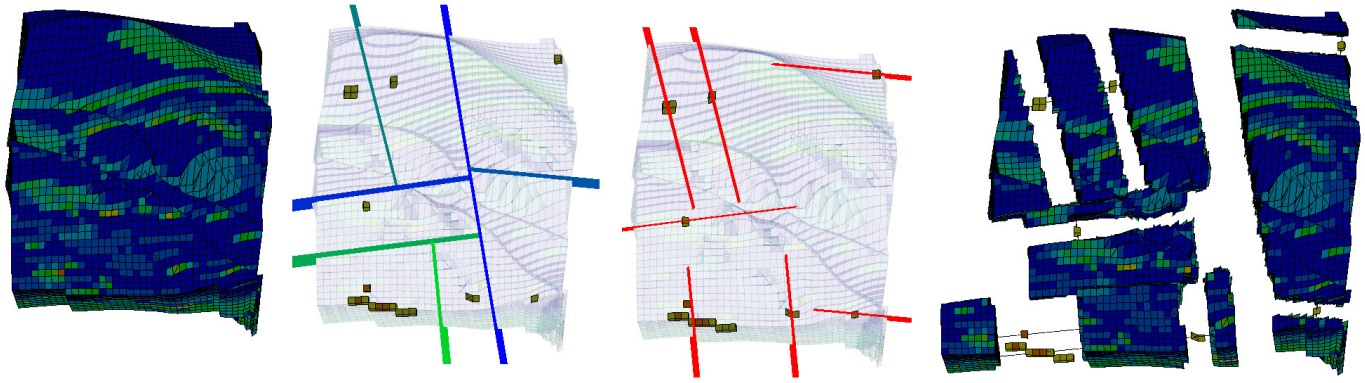

Fig. 9. Plane arrangement (from left to right): initial grid; 3D clusters within a subdivided space (non-extended planes); extended planes related to one or more 3D clusters, defining how they will be exposed; after the effect has been applied.

D. Extending the Explosion Tree

The second step performed by the data structure is the unprojection of all 2D planes from screen coordinate system to 3D object space, so in each node, 2D planes are replaced by 3D planes. So far, the data structure has a tree of nodes (representing planes) which subdivides the space in many regions and in an optimum case there is only one 3D cluster per region. Those planes, regions and their 3D clusters are depicted in the second image of Figure 9. Assuming that, for this example, their order of creation is represented by a color scale from blue to green, a correlation between those planes that perform the last level of division between the regions and those nodes which have less than two children can be made. Therefore, each of these planes are going to give birth to one or more new planes, which we call extended planes. The idea behind the transition between ordinary planes and extended ones is that while the former subdivides the space among regions, the latter lies within the region having the role of exposing 3D clusters within each region (see third image of Figure 9). A node which contains an extended plane is also called extended node, its plane normal is inherited by its immediate parent and its planes location is defined by the 3D resultant centroid of one or more 3D clusters (in the case there is more than one cluster per region). Summarizing this step, after unproject all 2D planes to 3D one, some nodes originate one or two extended nodes.

E. Calibrating the Explosion Tree

After defining extended and non-extended planes, the data structure is able to give the final direction of all grid cells (based on the normal vectors of the planes). The missing information though, is how much each plane is influencing each vertex position. To keep track of this information each plane has two offset values (one to each side). While the planes were calculated in a top-down approach, offsets are calculated from the bottom up to the root. First, the offset needed to expose the convex polygons in each extended node is calculated, and as extended nodes, they do not require information of any other node apart from themselves. As we cannot assume that convex polygons have a symmetric relationship with their immediate planes, two offset values are required, one makes the influence in the normal direction and the other in the opposite direction. Having all extended nodes updated, all non-extended nodes have their offsets calibrated based on their children where the final goal is to expose all convex polygons objects without creating any overlay between primary and secondary objects. All nodes have their offset calibrated based on the following steps:

- Through a post-order algorithm, each node (non-extended) takes in consideration its children by calculating how much offset those children will apply in its direction. For instance, if a child C of P has an offset α_C in its parent direction (i.e., the dot product between the two normal vectors is less than zero), the parent node's offset need to compensate with an offset of at least $\alpha_C \langle n_P, n_C \rangle$ in an opposite direction, where n_P and n_C are the normal of the planes P and C respectively. This is done for all non-extended nodes up to the root node.
- Still based on a post-order traversal, but instead of updating the parent nodes just based on their immediate children, it gets all nodes that belong to the paths that of the current node down to all its related leaf nodes. For each path it sums all the components which turns into its direction and compare with the current offset value. The current offset is updated only if the total sum is greater than its current value.

F. Obtaining the final position

After all previous steps, the final position of each vertex can be taken from the Explosion Tree. Either primary or secondary objects, our data structure gives the final position for all vertices in two levels. First, we use the non-extended nodes to calculate the final position for the primary objects and the partial position for secondary ones. The final position of the secondary objects is then calculated adding the offsets of the extended nodes. As a result of these steps, the exploded view diagrams effect can be achieved (see Figures 1 and 9).

G. Final Rendering

The meaning of final position is the place where all primary objects are exposed with no overlapping between either primary or secondary objects. To improve the visual correlation between primary and secondary objects regarding their initial and final positions, a line is drawn connecting the original convex polygon (centroid) position in each half of the respective secondary objects (see Figure 10). In addition, in order to provide a smooth transition our prototype animates the final effect by a linear interpolation between initial and final positions in ten steps (see Figure11).

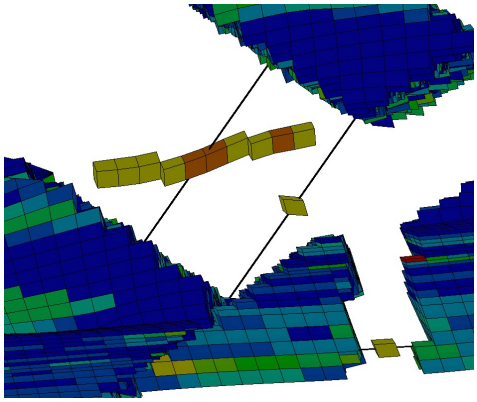

Fig. 10. Detail of the lines used to improve the visual correlation between primary and secondary objects.

III. Results and Discussions

We have used our prototype to successfully generate exploded views for corner-point grids (see Figure 12). Besides primary and secondary objects, the final result also depends on the view direction. We compute the time to build the tree and perform the effect to different models and for different primary/secondary object arrangements. Tests were performed with an Intel I5 CPU (2.53GHz) PC with 4GB RAM memory. In this work, the corner-point grids are real data currently used in the industry.

In order to evaluate the tree's performance, some measurements were taken and we concluded that the bottleneck to build the tree is the total amount of comparisons to find all planes. In a higher level, this amount of comparisons depends mainly on the gaze and the number of 3D clusters. For our typical case (i.e., up to 20 clusters), the time to build the tree is insignificant (less than 2 ms), so to evaluate this bottleneck we created artificial examples with larger numbers of primary objects. In the worst case in terms of number of comparisons, it took 10 ms to build a tree with 270 comparisons for a grid with 33,655 cells and 148 clusters.

Apart from the time to build the tree we have also measured the total time to achieve the effects. This time can be split into two parts, while the first one happens before building the tree, the second one happens after it. Our tree requires convex polygons as input and therefore, the time to convert grid cells into convex polygons might be taken into consideration. This is the first part and it is mainly influenced by the total amount of vertices of all 2D convex polygons. In this way, the screen resolution does not affect the performance as we are not dealing with pixels. After the tree is built, the system is ready to perform the exploded views effect to all vertices, therefore, the total number of vertices (including secondary objects) is the major factor that affects the performance in this second part. For instance, when applying the effect to a grid with 269,240 vertices, the time taken during the conversion (from primary objects to convex hulls) and to update all vertices are 370 and 90 milliseconds respectively. To finalize our workflow we use a ten-step interpolation between initial and final vertex positions, in our worst animation case, a grid with 228,056 vertices, a 37 fps animation was achieved.

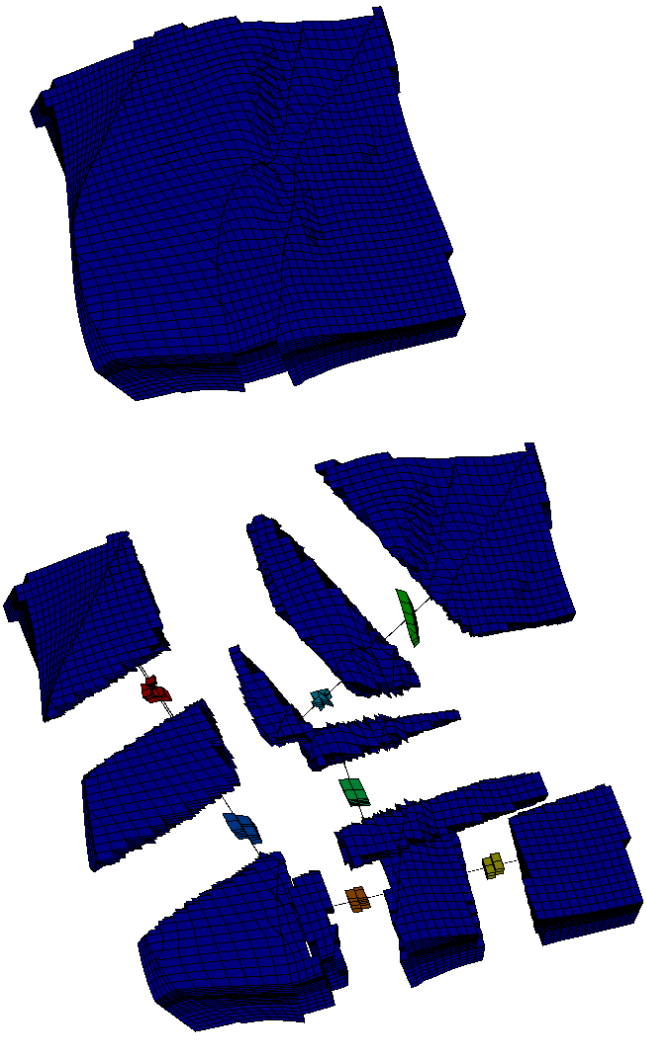

Fig. 12. Explosion View Diagrams: (top to bottom) original grid; seven clusters exposed.

The exploded diagrams effect was supported in an interactive time for all our grids. In our approach, we define planes to subdivide the 3D space, however when there is a large amount of 3D clusters it creates more empty space than would

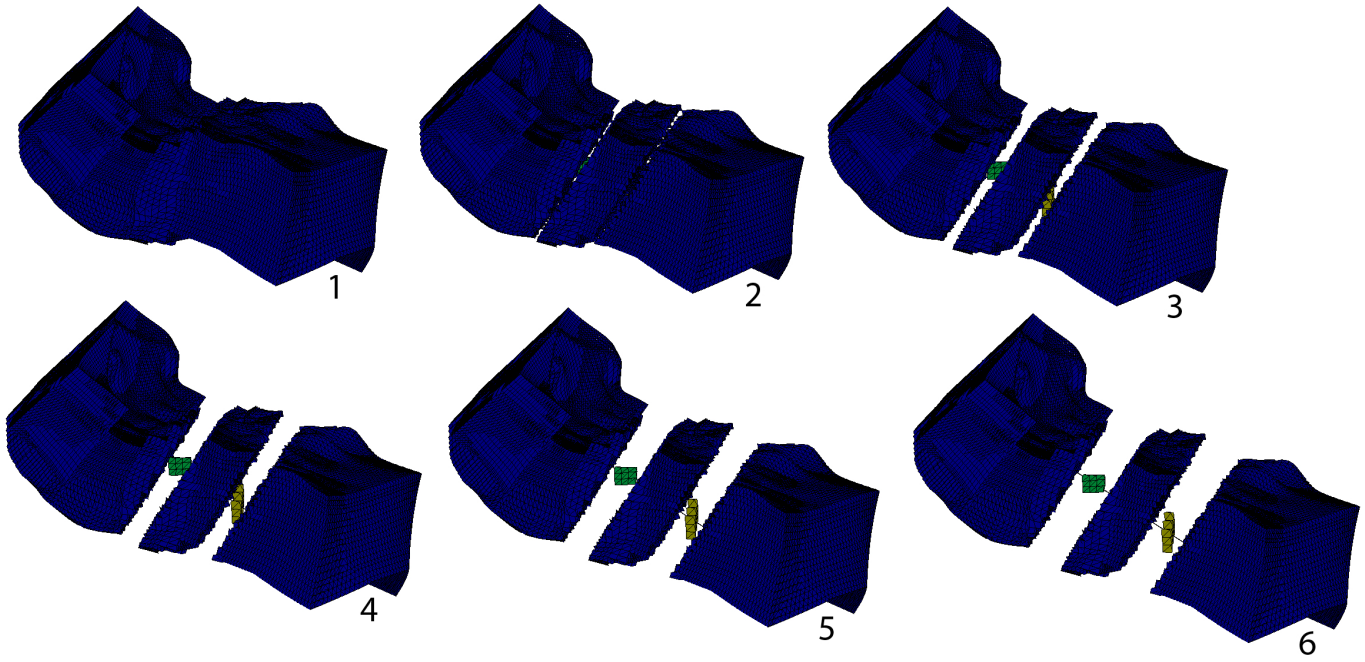

Fig. 11. Sequence of six animation steps (from a total of ten steps).

be required to just exposed the primary objects (Figure 13). A possible solution to overcome this limitation is the use of flexible surfaces to subdivide the space in a more optimized way. A functionality which might be useful to investigate the data is to apply the exploded diagrams from one gaze and varying the view angle to explore the model from different perspective (Figure 14).

IV. Conclusion

We have created a system and a data structure, *Explosion Tree*, to work with limited amount of information regarding the grid geometry and in dynamic scenarios. For some applications to consider the whole geometric information is too expensive. Using this data structure we reduce the amount of calculation, thus we do not compromise the interactive effects. Similarly, for dynamic scenarios, where primary and secondary objects are not the same all the time, we cannot rely on a pre-processing stage. As a consequence, each of the steps to build the tree need to be performed every time that the objects relationship or the view angle change, and in order to achieve the goal of having the effect in interactive time, a light data structure is required. To build the tree we assume that there is a collection of disjoint convex polygons, which are derived from primary objects. The main purpose of the tree is to guarantee that we can translate grid cells without any overlapping.

Further work is required in order to create a better calibration method wherein we can reduce the extra space between planes still maintaining that no overlap would occur. We intend to investigate different surface representations to subdivide the space in a more optimum way, we also plan to improve our data structure to support more general geometries, e.g., hexagonal and tetrahedrons. We are planing to incorporate an automatic optimum camera positioning given a set of primary objects and to allow properties to change in time. The second is a great challenge because we will have two different dynamics; one when the user change the properties-value range and another when the time steps of the properties change.

Acknowledgment

We would like to thank our colleagues for their useful discussions and advice. We also thank the anonymous reviewers for their careful and valuable comments and suggestions. This research was supported by the NSERC / Alberta Innovates Academy (AITF) / Foundation CMG Industrial Research Chair program in Scalable Reservoir Visualization.

References

[1] J. F. Thompson, B. K. Soni, and N. P. Weatherill, *Handbook of Grid Generation*. CRC Press, 1999.

[2] D. K. Ponting, "Corner point geometry in reservoir simulation," in *The Mathematics of Oil Recovery*. EAGE, July 1989.

[3] Y. Ding and P. Lemonnier, "Use of corner point geometry in reservoir simulation," *Society of Petroleum Engineers*, vol. 29933-MS, 1995, http://dx.doi.org/10.2118/29933-MS.

[4] Z. Martins, E. V. Brazil, M. C. Sousa, F. de Carvalho, and R. Marroquim, "Cutaway applied to corner point models," in *Workshop on Industry Applications (WGARI) in SIBGRAPI 2012 (XXV Conference on Graphics, Patterns and Images)*, Ouro Preto, MG, Brazil, August 2012.

[5] M. Luboschik and H. Schumann, "Discovering the covered: Ghost views in information visualization," in *Proc. Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2008.

[6] J. Diepstraten, D. Weiskopf, and T. Ertl, "Interactive cutaway illustrations," in *Computer Graphics Forum*, 2003, pp. 523–532.

[7] W. Li, L. Ritter, M. Agrawala, B. Curless, and D. Salesin, "Interactive cutaway illustrations of complex 3d models," *ACM Trans. Graph.*, vol. 26, no. 3, p. 31, 2007.

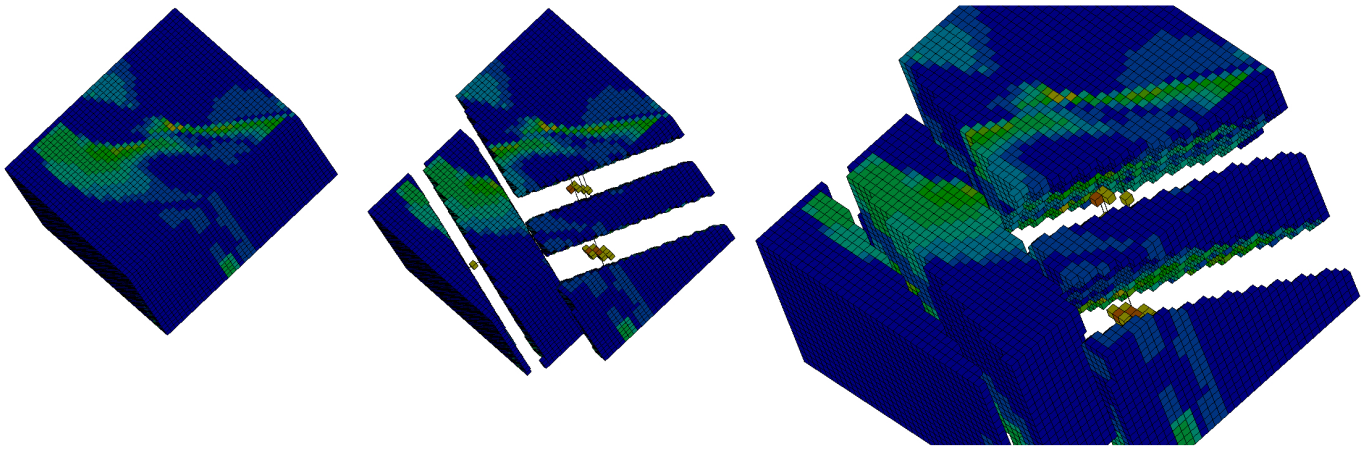

Fig. 14. Changing the perspective: (left to right) original grid; grid after exposing primary objects; the same effect from a different perspective.

[8] W. Li, M. Agrawala, B. Curless, and D. Salesin, "Automated generation of interactive 3d exploded view diagrams," in *ACM SIGGRAPH 2008 papers*, ser. SIGGRAPH '08. New York, NY, USA: ACM, 2008, pp. 101:1–101:7. [Online]. Available: http://doi.acm.org/10.1145/1399504.1360700

[9] M. Agrawala, D. Phan, J. Heiser, J. Haymaker, J. Klingner, P. Hanrahan, and B. Tversky, "Designing effective step-by-step assembly instructions," in *ACM SIGGRAPH 2003 Papers*, ser. SIGGRAPH '03. New York, NY, USA: ACM, 2003, pp. 828–837. [Online]. Available: http://doi.acm.org/10.1145/1201775.882352

[10] W. Li, M. Agrawala, and D. Salesin, "Interactive image-based exploded view diagrams," in *Proceedings of Graphics Interface 2004*, ser. GI '04. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2004, pp. 203–212. [Online]. Available: http://dl.acm.org/citation.cfm?id=1006058.1006083

[11] S. Bruckner and M. E. Groller, "Exploded views for volume data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1077–1084, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2006.140

[12] O. Karpenko, W. Li, N. Mitra, and M. Agrawala, "Exploded view diagrams of mathematical surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1311–1318, Nov. 2010. [Online]. Available: http://dx.doi.org/10.1109/TVCG.2010.151

[13] M. Tatzgern, D. Kalkofen, and D. Schmalstieg, "Compact explosion diagrams," in *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, ser. NPAR '10. New York, NY, USA: ACM, 2010, pp. 17–26. [Online]. Available: http://doi.acm.org/10.1145/1809939.1809942

[14] N. Elmqvist, "Balloonprobe: Reducing occlusion in 3d using interactive space distortion," in *In Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 2005, pp. 134–137.

[15] M. J. McGuffin and m. c. schraefel, "A comparison of hyperstructures: Zzstructures, mSpaces, and polyarchies," in *Proceedings of 15th ACM Conference on Hypertext and Hypermedia*, August 2004, pp. 153–162.

[16] J. A. Dennisson and C. D. Johnson, *Technical Illustration: Techniques and Applications.* Goodheart-Wilcox, 2003.

[17] A. Dumitrescu, J. S. Mitchell, and M. Sharir, "Binary space partitions for axis-parallel segments, rectangles, and hyperrectangles," in *In Proc. 17th Annu. ACM Sympos. Comput. Geom.* ACM Press, 2001, pp. 141–150.

Fig. 13. Limitation: large amount of 3D clusters might imply in more empty space than required. (top) original grid; (bottom) grid after the effect.