Review: 1  
**OVERALL RECOMMENDATION:**5  
**Evaluation Confidence:**4  
  
**Classification:**  
Research paper (presents innovative research results)   
  
  
**Summary:**  
The paper presents two extensions for optimizing A-buffer-based implementations to render combinations of semi-transparent meshes and volumes. The first contribution is an analysis of the depth complexity by counting the number of fragments per pixel in a first rendering pass. This information is then used to trigger a series of shader programs to adapt the size of the local buffer to the depth complexity of each pixel. The second contribution is a depth peeling approach to reduce the size of the local array so that it fits into the more efficient GPU cache and to avoid the rather slow global memory. The procedure is executed in a loop until all fragments are sorted.  
  
The general idea of the first contribution - the dynamic adjustment of the per-pixel memory requirements for sorting the fragments - is related to [MCTB12]. In [MCTB12], the optimal size of the local array is determined for each pixel with a rendering pass for fragment counting. Then, for each pixel, an individual index is computed to access a large continuous array (dynamic fragment buffer) and a second rendering pass then inserts the shaded fragments for local sorting and correct blending. Although the description of the algorithm in the presented submission is not totally clear, it seems that a single pass is sufficient to fill the fragment list and to compute the depth complexity. However, it is hard to judge how performance compares to [MCTB12] because filling the fragment list works differently. In [MCTB12], each pixel has its own access to an exclusive part of the global fragment list (dynamic fragment buffer) as opposed to the presented approach. Therefore, in terms of performance, it remains unclear if the ppAO method is beneficial because there is no direct comparison with [MCTB12] provided in the paper. Furthermore, ppOA employs a clustering of pixels of similar depth complexity to run a shader program with a constant array size for each cluster. In terms of memory requirements, the proposed method is not optimal as opposed to [MCTB12]. Altogether, the evaluation of ppOA is not (yet) convincing, but there is good potential that the new strategy might help increase the rendering performance on GPUs.   
  
The second contribution is the per-pixel depth peeling (ppDP) method to avoid overly large local arrays, which can become a bottleneck, if they do not fit into fast local memory (or cache) and need to be swapped into slower global memory. However, following the steps from B.a to B.e. in Section 5.3, I was not able to correctly sort a list of fragments. The description of the algorithm is too vague for reproduction. It looks like that a sliding window and some kind of insertion sort algorithm is used, but important details are missing in the paper. The results in Section 7 indicate that the authors successfully implemented an algorithm that achieves a noticeable speedup compared to a single iteration with one large array. However, without a clear description, it is hard to re-implement the algorithm. Judging the novelty of this method is therefore difficult.  
  
In summary, the paper has some novel ideas. In particular, dwelling on the analysis of depth complexity is a nice approach. The principles of both contributions are not new but rather incremental work. Nevertheless, the work has a good potential to have impact on the state of the art in semi-transparent rendering. The main weaknesses: the evaluation (in particular, a comparison to [MCTB12] needs to be included) and description that need to be improved. These changes might be doable in a minor revision, more realistically in a major revision.  
  
  
  
  
**Originality, Novelty:**  
6 - acceptable   
The two contributions (dynamic adjustment of the per-pixel memory requirements for sorting the fragments; per-pixel depth peeling) are much related to previous work and, thus, incremental. However, they show some level of originality.  
  
**Clarity of presentation:**  
4 - dubious - not quite acceptable   
The paper does not do a good job in explaining the important details of the novel parts, i.e., Sections 5.2 and 5.3, which take only one page. In fact, the rather dubious description significantly lowers the reproducibility of the paper. Instead, lots of space is wasted in Section 4, which basically explains an implementation of the A-buffer algorithm on the GPU.  
  
**Technical soundness:**  
6 - acceptable   
As far, as it can be understood from the technical description, the method seems to be technically sound. However: Table 1: The numbers/symbols (32, 64, 128, 128-8, 8\*, 16\*, 8\*, 8, %) of the columns are not described adequately and are misleading. The caption states the "no existing algorithm could render the scene without discarding unresolved fragments". Please be more precise. Section 8: "By applying our approach, it becomes for the first time possible, to fuse volumetric data with semi-transparent geometry which has a high depth complexity". Please be more careful with such statements: in [KGB\*09], volume and polygon data can be rendered with interactive performance and similar depth complexities.  
  
**Importance, utility:**  
6 - acceptable   
Efficiently fusing volumes and polygonal meshes is a relevant topic. However, the importance of the presented approach needs better demonstration. The results show that noticeable speedups are possible compared to a standard PPLL implementation, but the approach is not compared to a more state-of-the-art approach of order-independent transparency like [KGB\*09] or [MCTB12].  
  
**Suitable for event:**  
9 - excellent   
Certainly  
  
**Implementation:**  
4 - dubious - not quite acceptable   
I could not reproduce the steps in Section 5.3. to sort the fragment list so I think it would be quite difficult to implement the approach from the provided information in the paper.  
  
**Best paper award?:**  
NO   
  
  
**Completeness of References:**  
8 - very good   
Fine.  
  
**Additional comments for the authors:**  
For acceptance, the presented approach should be compared with a state-of-the-art technique that can handle dynamic memory allocation, e.g., [MCTB12]. I could imagine that the presented method is still beneficial, but there should be a direct comparison to meet EG standards. Furthermore, the description of the overflow optimization should be improved for full reproducibility. The results indicate that the technique has potential, but without more details it is hard to judge the value of the algorithm. In GPU computing, it is common practice to load chunks of data into fast shared memory, do some computations, and write the result back to global memory. Although the paper is not clear about this issue, I suspect that the ppDP technique is based on this principle. In this context, there should be a discussion of efficient GPU sorting algorithms.   
  
  
**Minor revisions required?:**  
YES   
  
  
**Revise & submit to CGForum?:**  
YES

Review: 2  
**OVERALL RECOMMENDATION:**6  
**Evaluation Confidence:**4  
  
**Classification:**  
Research paper (presents innovative research results)   
  
  
**Summary:**  
This paper describes an optimized a-buffer rendering approach for mixed transparent volumetric and geometric fused datasets. The proposed visualization algorithm supports efficient rendering of hybrid volume and geometry datasets by utilizing per-pixel depth peeling, per-pixel array optimization based on depth complexity histogram analysis. Specifically, the proposed approach utilizes depth complexity histograms (distribution of depth complexities across all pixels) of rendered scenes to bin regions of similar complexity for memory array optimization and specialized rendering for each complexity bin. They demonstrate a rendering increase of 3-8 times over current state of the art algorithms. They demonstrate a rendering increase of 3-8 times over current state of the art algorithms for relatively small datasets. The datasets tested with the presented algorithm are small compared to other previous studies (almost all less than 256^3). I strongly recommend applying the technique to larger data to demonstrate the performance benefits.   
  
With minor revision, this will be a good contribution to Eurographics 2014.  
  
  
  
**Originality, Novelty:**  
7 - good   
Novel optimizations and combinations on well studied algorithm  
  
**Clarity of presentation:**  
7 - good   
The arguments made in the exposition flow smoothly, however the following minor mistakes should be rectified:  
Section 2:   
“A common approach to mitigate the limit the inevitable performance…”   
“…, and should improve performance for a wide range if particular A-buffer implementations”  
Section 3:   
The abbreviation IOT should be OIT   
Section 4.1:  
“Each Fragment List is is responsible…”  
  
  
**Technical soundness:**  
7 - good   
  
  
**Importance, utility:**  
7 - good   
This will be utilized by many.  
  
**Suitable for event:**  
8 - very good   
  
  
**Implementation:**  
8 - very good   
Very good description of algorithms  
  
**Best paper award?:**  
NO   
  
  
**Completeness of References:**  
5 - marginal - only just acceptable   
Add references to earlier volume and a-buffer work (e.g., Ebert and Parent SIGGRAPH 90), other transparency volume/geometry combinations (Kniss, Hansen et. al), and alternative approaches  
  
**Additional comments for the authors:**  
The proposed approach builds upon the knowledge gained by the depth complexity histogram, and uses the concept of the A-buffer where fixed sized arrays are pre-allocated to all pixels. The A-buffers are used to store a sorted list of fragments per pixel for use in view-dependent blending. The challenge with the A-buffers, however, includes high memory consumption as fixed size buffers are allocated irrespective of the complexity of the scene at each corresponding pixel. Building on the A-buffer, the authors utilize the information provided by the depth complexity histogram for dynamic resource management by segmenting the image space into sections with similar depth complexity and optimal buffer sizes for each segment (the per-pixel Array Optimization, ppAO step). Next, the authors propose a new sorting procedure (the per-pixel Depth Peeling, ppDP step), where they break the sorting process into smaller pieces and not loading all pixel fragments at once. These two steps combined to tackle the inefficiencies caused due to high memory consumption constitute the major contributions of this work.   
The paper presents the new optimizations in enough detail for implementation and presents a nice comparison to a reference implementation. The example images and video show the results well. However, there appear some potential artifacts or unusual transfer function choice results in a couple of the animations – it is not clear if these are choices of the volume rendering transfer function of an issue with the algorithm. The rendering performance in the animation also seems to be jumpy and could be due to the different complexities of the views. This may be something the authors want to consider smoothing or setting a fixed (slower) rate so the user doesn’t find the variable speed of rendering annoying.  
The authors are also encouraged to update the references to include some more alternative approaches and more complete references on volume and a-buffer rendering.  
  
**Minor revisions required?:**  
YES   
  
**Revise & submit to CGForum?:**  
NO

Review: 3

**OVERALL RECOMMENDATION:**5  
**Evaluation Confidence:**3  
  
**Classification:**  
Research paper (presents innovative research results)   
  
  
**Summary:**  
The paper proposes to render multi-modal datasets using a-buffer based techniques running on a GPU. The novelty seems to reside in the a-buffer implementation that allows efficient visualization of transparent data.   
  
  
**Originality, Novelty:**  
5 - marginal - only just acceptable   
While the papers tries to address an important problem, and seems be successful in this task, the paper does not manage to convince me that the authors are doing a very good job there. I found the rendering examples to be "average" and unsupported by a user study, a few important components of the implementation not discussed, and quantitative results to lack explanations and analyses. It is however fair to day that they are successful at transparent rendering of fused/multimodal datasets, and so at a relatively interesting pace. I am not convinced however that handling and combining 128 depth layers is the right thing to do in visualization. Also a-buffer techniques have been around for ages so this is quite an iterative piece of work presented here.   
  
**Clarity of presentation:**  
4 - dubious - not quite acceptable   
The paper explains e few areas well, others are completely left out of the discussion. Figure 4 by itself, does show some control flow, but it quite inadequate to describe data flow for the implementation. For instance, having a clear step where buffers are clear is fine, but these buffers should be named there, and possibly illustrated as well. As such, it makes me it harder for me to understand the whole picture. A cuda implementation of the a-buffer seems to be the base for this work, but I don't think it is fair to refer just to the code and not to discuss the details of the CUDA implementation. It would be nice if the authors could be more precise on how data is handled, or deliver better, more explicit figures. I found figure 3 and 5 to be quite (too?) similar, and I think they should be merged inside the same figure instead of being in different pages.   
Results are not so clear to me. For instance the authors mention two different cards used for the test, but only give one rendering result (GTX580 I believe) in table 1. There is enough space there to accommodate two results, so I do not understand this at all. Table 1 caption should also explain precisely what columns are for, which is not the case. One column is about percentages, but the "unit" there is "X", not "%". Note that for "X", there is a specific symbol available that should be used. Also, I am not sure where this speed-up is coming from.  
The joint video is a little bit "too compressed" in my opinion. It would really benefit from a higher bitrate or using higher res. renderings just for the video and/or paper (512x512 being really borderline nowadays).   
Finally, I am not sure how the fragments produced from mesh, stream-line and volume data are blended to obtain the final colour. Maybe I have missed this in the text (which I doubt) but this should anyway be illustrated in a figure. Hence, more rendering details (not just data type) should be added to the beginning of Section 7.   
  
**Technical soundness:**  
7 - good   
Have no issue with this.  
  
**Importance, utility:**  
5 - marginal - only just acceptable   
I would suggest the authors to use more striking examples, and possibly prove the usefulness of the rendering produced with the system. This is not a strict requirement as the focus is more on interactivity, but I am not convinced that having so may samples combined in one pixel is useful.   
  
**Suitable for event:**  
7 - good   
This is an area fully suitable to the EG community  
  
**Implementation:**  
4 - dubious - not quite acceptable   
Many details are missing and should be discussed in the paper.  
  
**Best paper award?:**  
NO   
  
  
**Completeness of References:**  
6 - acceptable   
I think work has been on transparency (or not using transparency) that should be discussed in the paper. For instance, "Perceptually Based Depth-Ordering Enhancement for Direct Volume Rendering" discusses transparency and could very well be used to argue in favour of transparency here. Other approaches may however argue against the use of transparency when so many layers are used (e.g., "Instant volumetric understanding with order-independent volume rendering"). In other words, the authors may want to re-use a proven transparency model instead of using their own and have a bigger discussion on transparency.   
  
**Additional comments for the authors:**  
I think something needs to be done to make the paper a little bit more outstanding. A better description of the rendering pipeline is needed as well. 512x512 is a little bit small, but I guess beyond this, real-time is not possible anymore.   
  
  
**Minor revisions required?:**  
YES   
  
  
**Revise & submit to CGForum?:**  
YES

Review: 4  
**OVERALL RECOMMENDATION:**5  
**Evaluation Confidence:**5  
  
**Classification:**  
Research paper (presents innovative research results)   
  
  
**Summary:**  
The paper describes an approach for hybrid rendering of volume and geometry data using an adaptive A-buffer. The main contribution of the work lies in a careful analysis of performance bottlenecks with respect to current GPU architecture which results in an improved algorithm. The performance increase compared to previous approaches is significant.   
  
  
**Originality, Novelty:**  
5 - marginal - only just acceptable   
In terms of novelty, the main contribution of the paper is a rather technical one. While I am not aware of any specific approach that uses depth-complexity for optimizing A-buffer-based hybrid volume rendering, similar strategies have been applied to other scenarios. Nonetheless, the results are quite impressive and the presented technique may hence prove quite useful in practice.  
  
  
**Clarity of presentation:**  
5 - marginal - only just acceptable   
The paper is well-written but the description is somewhat difficult to follow due to the many abbreviations. The description of ppSP in Section 5.3 would also benefit from pseudocode to improve the clarity. Furthermore, illustrations would have been helpful.  
  
**Technical soundness:**  
7 - good   
As the analysis in Section 7.1 indicates, there benefits of the approach are heavily influenced by parameters which are dependent on the specific GPU architecture. A valuable addition would be an "auto-tuning" approach to automatically select good values using a set of benchmarks. In general, the analysis could have been more extensive with respect to GPU architectures (e.g. inclusion of AMD and Intel GPUs) and more detailed with respect to profiling results.   
  
**Importance, utility:**  
7 - good   
Combined rendering of volume data and geometry is important in several areas, and the technique presented in the paper achieves impressive speed-ups.  
  
**Suitable for event:**  
7 - good   
  
  
**Implementation:**  
7 - good   
  
  
**Best paper award?:**  
NO   
  
  
**Completeness of References:**  
7 - good   
The discussion of related work is adequate.  
  
**Additional comments for the authors:**  
The main concept used in the paper to achieve the substantial performance increases is to adapt the storage scheme to the per-pixel depth complexity. This is then used in two ways:  
  
a) the size of local arrays is adjusted (using specialized shaders) for rgeions in screen space which have similar depth complexity (per-pixel array optimization, ppAO).  
b) the use of "local depth peeling" (ppDP) to allow for arbitrary depth complexity while still using fast local arrays.  
  
Both approaches result in a more optimal use of GPU resources and allow the rendering of scenes with a high depth complexity at good frame rates. However, they are also quite technical and do not represent major conceptual innovations.   
  
Hence main strength of the paper is the practical improvement in performance, but it is rather weak in terms of scientific novelty, and I am not sure whether this is enough for a EG paper.   
  
In summary, I am split about this paper and would not strongly argue in its favor, but I also think that it does contain a nice (albeit limited) contribution. However, the presentation should be improved and the paper needs a more detailed performance analysis (see above).   
  
  
**Minor revisions required?:**  
YES   
  
  
**Revise & submit to CGForum?:**  
YES

Review: 5  
**OVERALL RECOMMENDATION:**4  
**Evaluation Confidence:**5  
  
**Classification:**  
Research paper (presents innovative research results)   
  
  
**Summary:**  
The paper describes an A-buffer based algorithm for order independent transparency that relies on a depth complexity histogram (DCH) to estimate the number of samples to be stored per pixel. Using the DCH of a given frame, the proposal is to divide the image into regions with similar depth complexity, thus providing tigther and lower bounds to the number of samples to be stored. The paper shows performance results for 4 different datasets, but did not provide any comparison against competing strategies, only a basic A-buffer reference implementation.  
  
  
**Originality, Novelty:**  
6 - acceptable   
The use of a depth complexity histogram to estimate the size of per-pixel storage requirements for an A-buffer structure has not been presented in the literature before. The fact that it relies on a single frame reduces its usage to this frame, since other views might generate different DCH with other behaviours. Although the idea is new, I do not think it was fully validate and explored.  
  
**Clarity of presentation:**  
5 - marginal - only just acceptable   
I prepared several comments regarding the text, they were posed in the section "additional comments"  
  
  
**Technical soundness:**  
4 - dubious - not quite acceptable   
There are many questions that need to be answered, they were posed in the section "additional comments"  
  
**Importance, utility:**  
4 - dubious - not quite acceptable   
The proposal has merit, the idea of using a depth complexity histogram to estimate memory usage in OIT algorithms is interesting. However, the paper has 2 serious shortcomings. There is a lack of analysis of the impact of the DCH in different viewpoints (how they vary) and how the algorithm can cope with these changes. Also, for the proposal to be competitive, it need to be compared agaisnt the state of the art A-buffer based OIT algorithms.  
  
**Suitable for event:**  
4 - dubious - not quite acceptable   
The paper topic is very interesting and suitable for the event, but the current format I think the paper it is not quite ready.  
  
**Implementation:**  
6 - acceptable   
There are minor comments raised before but overall the approach can be reproduced.  
  
**Best paper award?:**  
NO   
  
  
**Completeness of References:**  
7 - good   
The references are ok.  
  
**Additional comments for the authors:**  
Abstract:  
- “In many cases, only geometric and volumetric data sets together describe a single phenomenon under observation.” >> suggestion: “… sets together are able to describe…”  
- I agree that your work analyzing data sets very is important to the whole algorithm development, but the abstract is not the place for it. Please, go straight to the point: what problem are you solving? What advantages does your solution bring? Why it is better than previous solutions? A reader must finish your abstract with a clear idea of what the paper shows.  
- At this point I understand that this paper is proposing a rendering algorithm for hybrid data representation. But I don’t have any clue of how it works, or its advantages. And, later on the introduction, you claim your proposal is a visualization algorithm. Please clarify the nature of your proposal.  
  
Introduction:  
- I am confused by what you mean by “incorporation of transparency effects in volumetric rendering can be considered as an integral part”, please, clarify.  
- “imaging dependent areas of science.” >> imaging-dependent areas of science.  
- “The fundamental challenge arising when…” >> The fundamental challenge, arising when…”  
- “which enables us to annul the memory limitations of modern GPUs,” >> the term “annul” is too strong. Only unbounded physical memory could “annul” memory limitations (the amount of fragments generated during rendering may exceed the GPU physical memory even before the stage where your algorithm can be applied).  
- What is the purpose of Figure 1, since it isn’t even referred?   
  
Related Work:  
- “To ensure correct blending order or multiple semi-transparent samples,” >> To ensure correct blending order of multiple semi-transparent samples,  
- Here you say “In this work we focus exclusively on the A-buffer approach..”, and on section 5.1 you say “we propose a new sorting procedure, called per-pixel Depth Peeling”. This is confusing. Please, review your terminology.   
- “The A-buffer concept… the concept has spawned multiple similar techniques and variations…. Everitt [Eve01] present one of the first hardware adaption,” >> as far as I know, the solution proposed in Eve01 is clearly a depth peeling approach, very different from the A-buffer.  
- “The main goal works cited above is to minimize this memory consumption.” >> The main goal of the works cited above is to minimize this memory consumption.   
- Please, state (in the related work section,) the reason why you choose the A-buffer variation using PPLL instead of the one using DFB.  
  
Visualizing Scenes With High Depth Complexity  
- Typo: “IOT” >> OIT  
  
Depth Complexity Histograms (DCH)  
- “We define the term Depth Complexity Histogram (DCH) to a histogram that plots the distribution of depth complexities across all pixels in an image.” >> This definition is incomplete. Please rephrase.  
- Figure 3 appearing before Figure 2 is very disturbing for the reader. Please, correct the ordering the figures appear.   
- “The computation of DCH is straightforward.” >> Unnecessary.  
- “First, a gray scale image is rendered where the pixel color is defined solely by the number of fragments for each pixel.” >> How does the number of fragments define a color? What is the color range? [0 ,255]? Does it mean that your approach cannot handle more than 255 transparent layers?  
- Figure 3: what does black mean in the left figure? And what does white mean? The background seems to present a degrade, what does it mean?  
The histogram in the right image is from the frame shown in the left or from other frame of the same scene?   
- Figure 2 lacks explanation of what the figure is presenting. What each image means, how this information is used…   
  
Visualization Challenges for Data Fusion  
- How many “commonly occurring scenes” where analyzed? What is the relevance of the chosen scenes?   
- What happens when you zoom in the scene such that all pixels present high depth complexity?  
- Figure 2.a and Figure 3 left do not seem to be created from the exact same scene. Is Figure 3 simplified from Figure 2.a?  
- Why is PPLL (or even DFB) “dependent on a maximum number of supported depth layers”, since it can handle unbounded amounts of memory?   
- What is a “bad pixel”?  
- “The challenge therefore remains to create an approach that fully utilize advanced parallel architectures and fast, but less flexible, memory hierarchies even for dynamic scenes with rapidly decreasing DCHs.” >> I cannot understand this paragraph; please, rephrase.   
  
Semi-Transparent Data Fusion  
- “gain significant performance increase.” >> redundant.  
- In the second paragraph, please, clarify that the hardware architectures are from GPUs, not CPUs.  
  
Adaption to Modern Architectures   
- Please, clarify that the hardware architectures are from GPUs, not CPUs.  
- What is a “per-pixel anchor”?  
- How is decided the size of the Fragment Pool?  
- Why does the atomic counter must have 64 bits?  
- Clarify that the description given in this section refers to PPLL and not to FFB neither DFB.  
- Doesn’t the Fill Step occur during the rasterization of fragments (not after)?  
- What does it mean to resolve the Local Array?  
- Were your solution implemented in CUDA or within a graphic pipeline?  
- Figure 4: Why there are 4 arrows leaving the ppAO stage to two different stages? Is this a non-deterministic algorithm?  
  
Dynamic Depth Complexity Management  
- What are the limitations of the current hardware?  
- What are the two novel optimizations?  
- “We begin by outlining the full algorithm before providing additional detail to each part.” >> Remove.  
  
Depth Adaptive A-buffer Rendering  
- Typos: “Step step”  
- How is defined the optimal buffer size for each segment?  
- Move this text under section “Dynamic Depth Complexity Management”  
  
Dynamic Resource Management Using Per-pixel Array Optimization (ppAO)  
- “Per-Segmented Resolve:” >> Per-segment Resolve?  
- “although it sounds intimidating” why does the segmentation step sounds intimidating?  
- How is the correct shader program (the one with the correct local array size) launched for the Per-segmented resolve?   
  
Preventing Local Array Overflow Using Per-pixel Depth Peeling (ppDP)  
- “the Resolve Step of the A-buffer procedure in Section 4.1 is modified to perform the following steps (executed in a loop), as illustrated in Figure 4: ” >> I couldn’t find the loop in Figure 4, please, point it out.  
  
Implementation Details  
- I suppose that fragment lists from rendering of different data representation are generated separately and appended before the ppAO step. But I could not confirm it in the text. Please, clarify this issue.  
- “During execution, the fragments are resolved in order.” >> during the execution of which step? And in what order are the fragments resolved?  
  
Results  
- I would like to see images for each one of the separated datasets that compose one scene.  
  
Performance Comparison  
- Why the need for reporting test in two GPUs with the same architecture? And, still, only presenting results for one of them?  
- “..This notable increase in performance for shorter array sizes is most likely a product of driver optimization, and its source being the limited size L1 cache.” >> Split the sequential processing of an array of size N into two parallel sequences of size N/2 suggests that the performance gain comes from the high GPU parallelism, not from the driver.  
- Table1: what is the maximum depth complexity for each scene? i.e. does not make sense to present results for a local array size of 128 for a scene with max. depth complexity of 64 (it is misleading). What do the “\*” mean? What is the “%” column?   
- I would like to see comparisons with algorithms other than PPLL (e.g. FFB).  
  
Conclusions and Future Work  
- You claim to have presented a visualization technique, but what I saw was an increment over a known OIT rendering technique (PPLL).  
- “By applying our approach, it becomes for the first time possible, to fuse volumetric data with semitransparent geometry which has a high depth complexity” >> Why cannot PPLL handle the fusion?  
- Your conclusion is not a conclusion; it only sums up all that you already said before.   
- “In the future, we would like to explore this and other interesting possibilities.” >> remove.  
  
  
**Minor revisions required?:**  
YES   
  
  
**Revise & submit to CGForum?:**  
NO