Hierarchical Plane Extraction (HPE): An Efficient Method For Extraction Of Planes From Large Pointcloud Datasets

Author 1:
Naveen Anand Subramaniam,
Project Assistant,
Living Environments Laboratory,
Wisconsin Institute for Discovery.

Author 2:
Dr.Kevin Ponto,
Assistant Professor,
Living Environments Laboratory,
Wisconsin Institute for Discovery.

Search Keywords: HPE, plane extraction, 3D pointcloud, hybrid rendering, planar models, RANSAC
Category: Big Data

Description: This paper describes a fast and efficient algorithm to obtain planar models from high density 3D pointclouds using spatial hashing.

Abstract

Light Detection And Ranging (LiDAR) scanners have enabled the high fidelity capture of physical environments. These devices output highly accurate pointcloud datasets which often comprise hundreds of millions to several billions of data points. Unfortunately, these pointcloud datasets are not well suited for traditional modeling and viewing applications. It is therefore important to create simplified polygonal models which maintain the original photographic information through the use of color textures.

One such approach is to use RANSAC plane detection to find features such as walls, floors and ceilings. Unfortunately, as shown in this paper, while standard RANSAC works well for smaller data sets, it fails when datasets become large. We present a novel method for finding feature planes in large datasets. Our method uses a hierarchical approach which breaks the dataset into subcomponents that can be processed more efficiently. These subsets are then reconfigured to find larger subcomponents, until final candidate points can be found. The original data set is then used along with the candidate points to generate final planar textual information. The Hierarchical Plane Extraction (HPE) method is able to achieve results in a fraction of the time of the standard RANSAC approach with generally higher quality results.

Introduction

LiDAR scanners are widely used to scan physical spaces such as a room or a house to obtain 3D pointcloud data. Each point contains data for the spatial XYZ coordinates and the RGB color. As LiDAR scans can output high resolution scans consisting of more than 10 million points for a room and almost a billion points for an entire house, the file sizes of these datasets may vary from 1GB to even 20GB. In this regard, processing, manipulating visualizing the

entire pointcloud can be extremely difficult.

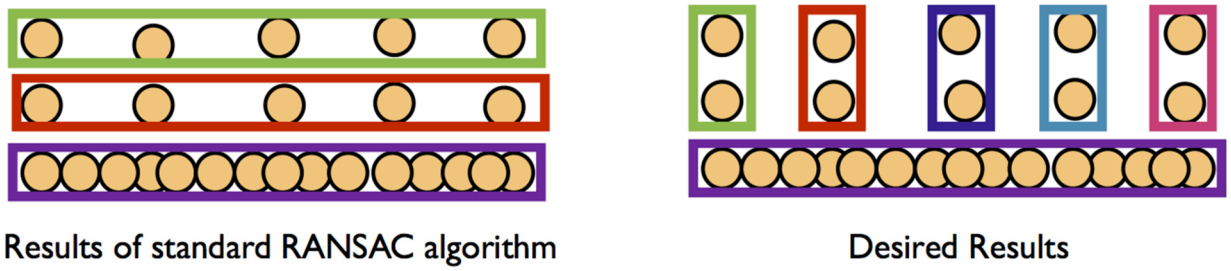
One approach to solve this issue is to find points which represent planar segments and convert this information into 3D geometry and textures. This format is generally how 3D information is stored for architectural models and video games and, in turn, this representation is what computational hardware is optimized for.

One of the conventional methods for finding a plane is RANdom SAmple Consensus(RANSAC) which works as follows:

1. Randomly select three points from the pointcloud and calculate a plane equation between them.
2. Check all other points to see if they fit in the plane equation within the threshold distance specified.
3. Get an estimate of how many points fit in the plane found which are the inliers.
4. Find a different plane equation by selecting a different combination of three points.
5. Repeat steps 2 to 4 until the specified maximum iterations allowed.
6. The plane with the most inliers is the best match.

While other methods can be used to find planes through pointcloud data, such as Hough Transforms, RANSAC has been shown to be most effective (Tarsha-Kurdi, 2007). Unfortunately, RANSAC is not without limitations. The method evaluates models on all points in the dataset and the percentage of points that are a fraction of the original dataset is measured. This process is iteratively repeated until the model with most inliers is chosen. Although RANSAC can find a model in a dataset with a lot of noise, the number of samples required and the time taken for the process is substantially high (Raguram, 2008).

The algorithm works best when finding a single plane in the dataset while ignoring the noisy non-planar point around it. When finding many planes in a single data set, a "best" plane must be determined. This is often done using a plane with the greatest number of inlier points. As shown in *Figure 1*, this can produce results in which large sparse planes are favored over smaller denser planes. This can result in small walls being missed in favor of large sparse floors as shown in *Figure 2*.



(a) (b)

Figure 1. (a) shows pointcloud having one horizontal plane(floor) and five vertical planes(walls) perpendicular to the horizontal plane. Standard RANSAC is designed to obtain the planes with higher inliers and thus obtains three parallel horizontal planes. (b) shows the correct extraction of planes when the normal values of the points are also taken into

consideration while extracting planes.

One approach to try to overcome this problem is to account for other factors in the dataset. For the example of planes, PCL (Rusu, 2011) can utilize normal information in the RANSAC. Unfortunately, calculating normal information on a point-by-point basis often utilizes neighborhood information which can be a costly process in terms of time.

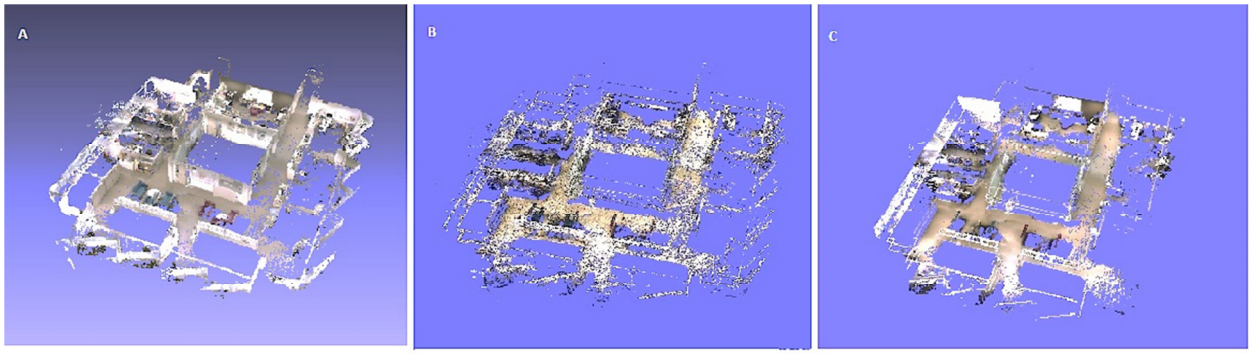


Figure 2. A) shows the original pointcloud with 471829 points B) shows the 10 planes extracted by RANSAC with 5mm distance threshold. The extracted planes demonstrate the issues described in Figure 1. C) shows 10 planes extracted by HPE method with 5mm distance threshold.

The simplest solution to this problem would be to subsample the dataset until the normal estimation and plane detection algorithms could run effectively. However, simply forcing points to a grid causes loss of information, discontinuities in the dataset and leads to suboptimal results. Therefore we want to generate an algorithm which is able to process these large dataset without removing information.

We present our algorithm, Hierarchical Plane Extraction(HPE) which is able to create textured planar geometry from extremely large pointcloud datasets. HPE builds up a hierarchical series of planes from subsections of data until large planes can be found which encompass the entire data set. Texture information is generated from the original dataset, enabling high-resolution textures to be created. The algorithm is shown to produce generally better results in shorter amounts of time compared to the existing RANSAC approach.

Previous Work

Finding models from unstructured point clouds is a concern that has been addressed since the origin of pointcloud data (Varady, 1998; Vosselman, 2004; Schnabel, 2007). This not only provides higher levels of abstraction of the data but also reduces the effective size of the data which can be used for visualization by hybrid rendering of models and points. By compressing the numerical information about the points by algorithms that provide the geometries of the points and by controlling their level of details, visualizing pointclouds could be made effective and interactive (Remondino, 2006).

To do so, a time efficient algorithm that uses very less points as samples from the original pointcloud is intuitively the best solution. Finding an initial solution and attempting to enlarge the model along with points fitting in the plane (Fischler, 1981) would work for a small dataset. Although since all the points are considered valid solutions this would mean both signal and

noise equally contend to form models. There is also a maximum number of attempts that are allowed before finding the best possible solution. This would be unimaginably high for a big dataset which is time consuming. Choosing a lower number of attempts allowed would mean the probability of missing the best possible solution decreases (Fischler, 1981; Tarsha-Kurdi, 2007).

The 3D Hough transform estimates planes by calculating a transformation matrix H and detecting peaks in it (Hough, 1962). Since this method is not only time consuming but also determines planes statistically without taking into account their significance in the original pointcloud data, it is shown that the RANSAC method determines planes much faster and accurately than the 3D Hough transform (Vosselman, 2001; Illingworth, 1988). In addition to using the standard RANSAC, it is shown that it is necessary to use the standard deviation of the inliers that form the plane to estimate the best solution (Tarsha-Kurdi, 2007).

Although the standard RANSAC fetches appreciably faster and better results than the 3D Hough transform and the standard deviation values could be used, it also obtains points without considering the orientation of the plane in the original unstructured pointcloud data. To differentiate points that do not fall on the same plane, the normals of the points could be used along with the spatial location of the points during the process (Schnabel, 2007; Hoppe, 1992). Since the pointcloud obtained from a LiDAR scanner does not have the normal information associated with it and calculation of normals for all the points on such a huge dataset is time consuming, it is almost inefficient to use standard RANSAC in the beginning.

By sampling a few points and finding planes and then superimposing on the original point cloud data, it is shown that we could reduce the time taken to estimate planes from the pointcloud (Tarsha-Kurdi, 2007). To use this method, sampling a few points from only from signal and excluding noise from the input data is crucial. The process of eliminating noise from signal has been of concern and has led to adaptations of RANSAC that needs preprocessing steps to sample signal points before running standard RANSAC (Chum, 2002; Clarke, 1996). Others have used Octree structures to improve the standard RANSAC implementation (Meagher, 1982; Frisken, 2002; Elseberg, 2011). Usage of Octrees while also not losing the edge points has also lead to significant results over standard RANSAC (Woo, 2002). Sampling through iterative simplification while not losing the shape of the surface has been done and proved to be fast and robust (Pauly, 2002).

Three-dimensional points can be spatially hashed using the hash dimension of every voxel and the number of hash tables available. It is shown to be an effective algorithm that splits the points in the datacloud into voxels in order of linear time and also eliminates collisions (Teschner, 2003).

The HPE method puts forward such an algorithm that samples signal from the input data, finds planes, uses the information to superimpose on the original pointcloud data to form planar models of the original pointcloud. These planar models can be used in conjunction with the non-planar points in a hybrid renderer to improve the rendering performance (Wahl, 2005; Chen, 2001; Wu, 2005).

HPE: Plane Extraction by Multi-Level Hashing

Dividing the Data Set

The goal of the HPE approach is to convert one large complex problem into a series of small simple problems. This divide and conquer method requires the data to first be split into a series of smaller pointclouds based on a voxel grid. As the data sets were extremely large, allocating a voxel grid was not feasible. Tree based approaches (such as octrees) break the data into subsets, but store a hierarchy of information that was extraneous for our application. Therefore, we chose to use a spatial hash approach. Spatial hashing has the advantage of creating a fixed memory footprint and can be computed in O(n).

The points are divided by using a spatial hash function (Teschner, 2003) to split the big data set to many small data sets. To hash the data set appropriately there needs to be knowledge of an optimal hash dimension value and the number of hash tables. Both the values are purely dependent on the data set that is under consideration. The number of hash tables is presumed to be of an initial value depending on the number of points in the data set. The initial value of the number of hash tables is presumed to be a power of 10 based on the size of the input point cloud data. It is generally two orders of magnitude lesser than the size of the input point cloud data to accommodate around 100 points per voxel. Having a large number of hash tables would increase the time taken to process the voxels while having a small value would cause collisions when spatially hashed.

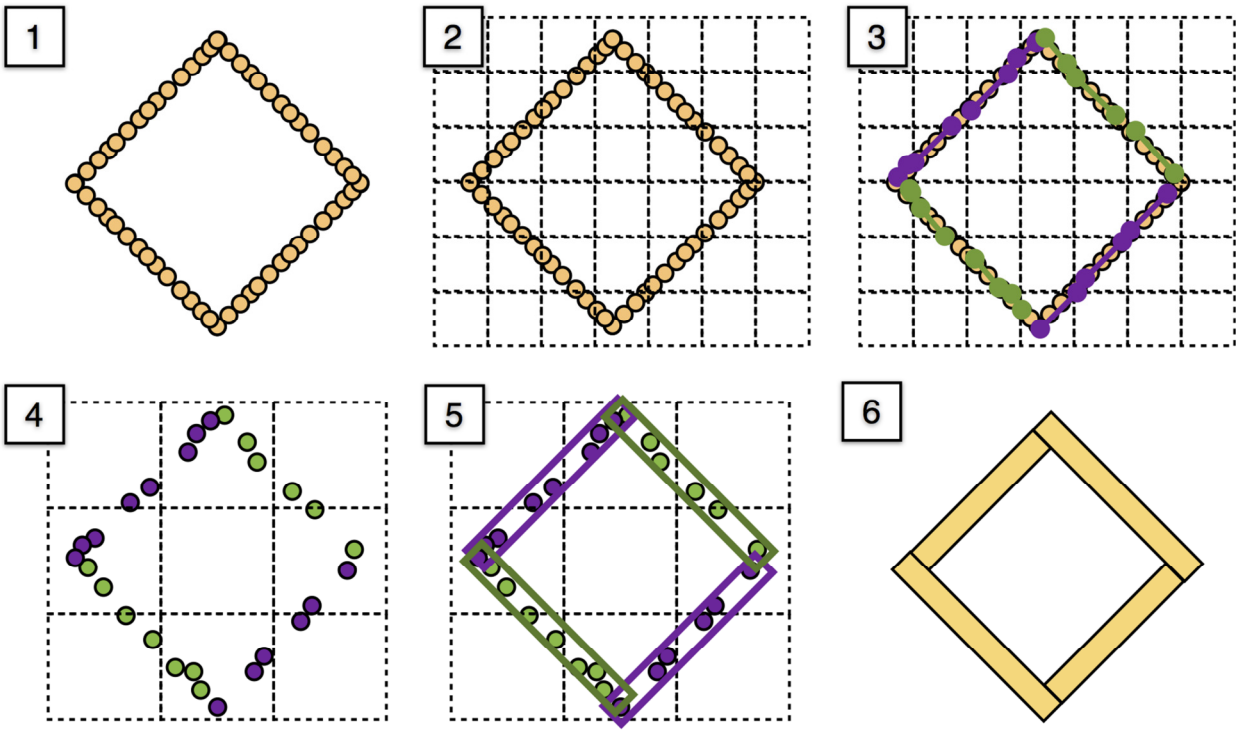


Figure 3. (1) Input point cloud. (2) Segmenting into 42 small pointclouds of 5 average points through spatial hashing. (3) Obtaining planes in every small point cloud by standard RANSAC and estimating their normals (shown in purple and green). (4) Increasing the hash dimension iteratively until a few candidate points with normals are obtained. (5) Plane estimation in candidate pointcloud through standard RANSAC shown as bounding rectangles. (6) Obtaining

all planar points by superimposing the bounding box on the original pointcloud and drawing textures.

The hash dimension is recursively calculated until the average number of points in every hash lies between a minimum threshold and a maximum threshold. The minimum and maximum value for the average points per hash were empirically determined to be 50 and 200 after considering the trade-offs between the time taken to detect planes in a hash and the number of hash tables. During the estimation of the hash dimension, if the number of hash tables is not big enough to fit less than the maximum threshold value, the number is increased until the average points per hash fit between the thresholds. The hash dimension is usually a very small value in the order of a few centimeters for dense pointcloud data. Further, there is a collision check in every voxel after hashing to ensure they do not have points falling from a different voxel into them.

Plane Extraction on the Divided Data Sets

The divided data sets have an average points per hash value in terms of a few tens or hundreds. This is a very small and efficient number for RANSAC to find planes. Standard RANSAC is iteratively run on every voxel to find one or more planes. To keep the points to a minimum without shifting the plane equation, we save only the four boundary points of the plane found along with the normal value of the plane. The normal value of the plane is almost effortlessly calculated now as the plane equation is already available. Thus all the XYZRGB points in the voxel are reduced to just a few planar XYZNormal boundary points. If no planes could be estimated, no boundary points are saved from that voxel.

From these boundary points that are few in number, we obtain the bigger planes and their boundary points by repeating the same hashing on the normalized boundary points but after selecting a bigger hash dimension than the previous iteration. Neighboring boundary points now fall into the same voxel since the hash dimension is increased. As we have the normal values along with the points, inliers are calculated not only if a point falls within a threshold distance from the plane equation but also if the point's normal aligns with the plane's normal. This ensures that intersecting planes are accurately identified as different planes even though the points of one plane fall within the distance threshold of another plane. The new boundary points are saved along with the normals from the plane equation. This technique enables us to find bigger and bigger planes and consequently reducing our boundary points with every iteration.

This process of growing in hash dimension is repeated until the sum of all boundary points in all the voxels comes down to a manageable value for running a final RANSAC. At the end of this process, there are a few meaningful points with normals that represent different planes in the original data set. A salient feature of this technique is that as the boundary planes are saved for all voxels, bigger planes in the original data set will fall into many voxels after hashing spatially and thus will have more boundary points representing them in the final set of candidates. Smaller planes in the original data set will have lesser boundary points as they fall into lesser voxels. This gives the advantage of selecting the biggest plane first from the candidate pointcloud.

After identifying a set of candidate boundary points with normals, RANSAC is run on the

whole candidate pointcloud to estimate planes. We obtain the plane with the most boundary point inliers. The maximum and minimum spatial coordinates of that plane are determined thus yielding a boundary box of the plane. Using these maximum and minimum spatial coordinates, all the XYZRGB points from the original point cloud that fall within these coordinates are extracted. The remaining points from the boundary pointcloud and original pointcloud are used for the next iteration of extracting planes. The process is repeated until a stopping condition that may be specified in terms of number of planes obtained or number of points left.

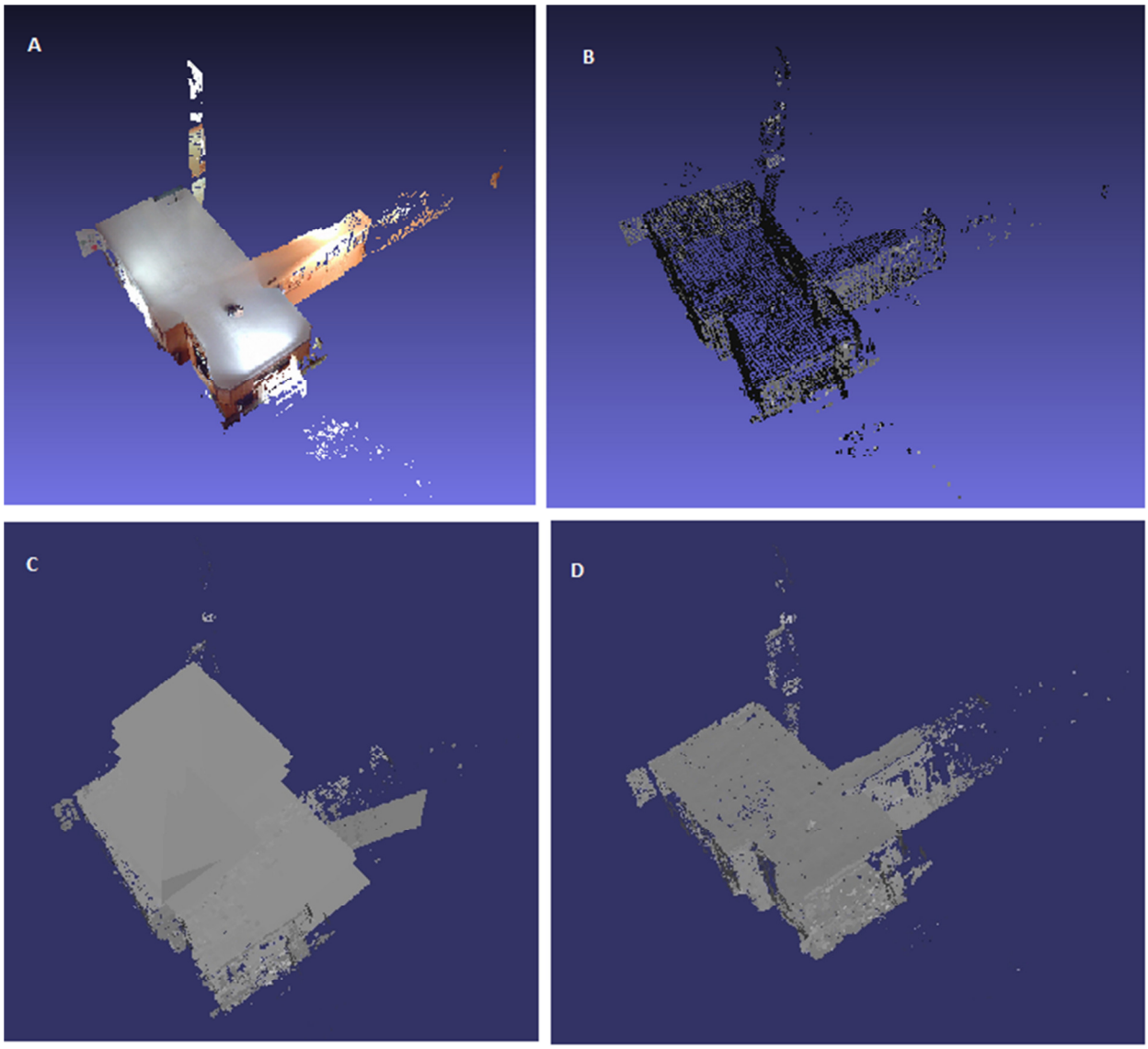


Figure 4. A) Image showing original point cloud of a room containing of 10.7 million points. B)Image showing boundary points of the planes found in a small hash dimension along with their normals of the plane (Points shown as varying from white to black depending on their normal angle). C) Image showing planes found in voxels with a small hash dimension without eliminating collisions. d) Image showing planes found in voxels with a small hash dimension after eliminating collisions. The model looks to be well bound, consistent and retains the shape

of the original point cloud data.

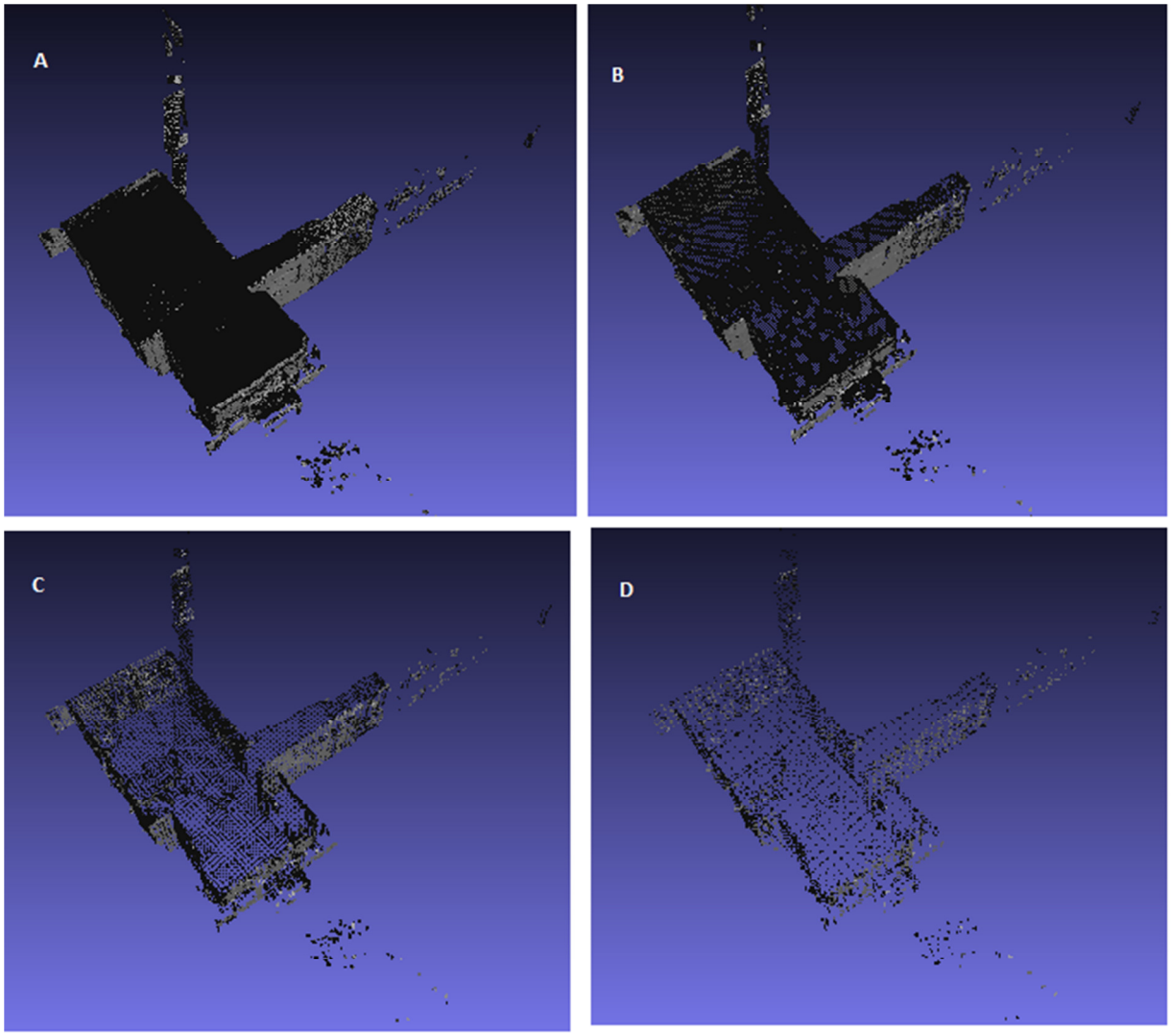


Figure 5. Images showing the hash dimensions growing and hence the boundary points reducing meanwhile maintaining the shape of the original pointcloud.

Results

As mentioned earlier, the two key objectives of the HPE method was to fasten RANSAC while also maintaining the quality of the planes extracted. The HPE method estimates the planes from a candidate set of points that are a subset of the original point cloud. Since the number of candidate points is in the order of a few thousands and ten-thousands, the planes found were accurate even for a lower value of allowed iterations. For the same reason that the HPE estimates planes from a smaller subset of points, it was found to be much faster than the standard RANSAC. The results were found to be distinctly faster for larger pointclouds when using the HPE method.

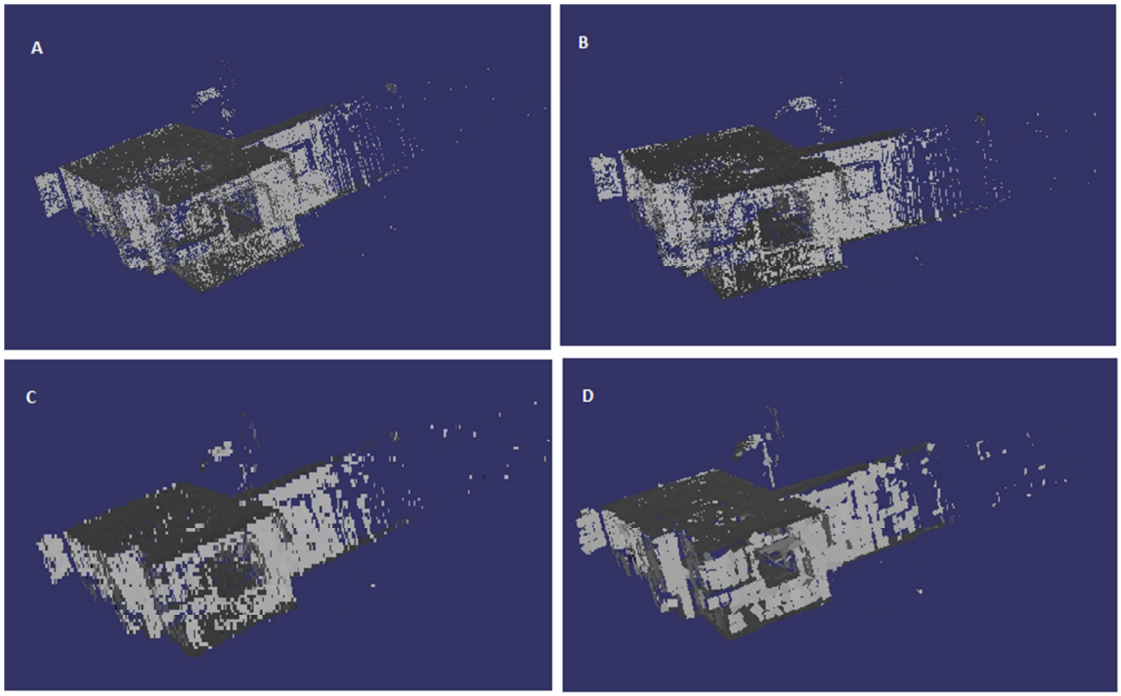


Figure 6. Images showing the planes found in every voxel as the hash dimension grows larger.

POINTS IN INPUT FILE	NUMBER OF PLANES FOUND	METHOD	TIME TAKEN
0.4 Million	10	Standard RANSAC	64m 23s [1]
		HPE	11m 32s
10.0 Million	10	Standard RANSAC	10m 31s
		HPE	3m 1s
10.0 Million	10	Standard RANSAC	11m 31s
		HPE	23m 13s
10.8 Million	20	Standard RANSAC	Bad Output[2]
		HPE	9m 8s
27.6 Million	10	Standard RANSAC	Bad Output[3]

[1] Output shown in Figure 2B) for 10 extracted planes and 1000000 allowed iterations.
[2] Output shown in Figure 7A) for 20 extracted planes and 1000000 allowed iterations.
[3] Output shown in Figure 7B) for 10 extracted planes and 1000000 allowed iterations.

		HPE	14m 10s

Table 1. Results showing comparison of time taken to extract planes using the standard RANSAC algorithm and using the HPE method.

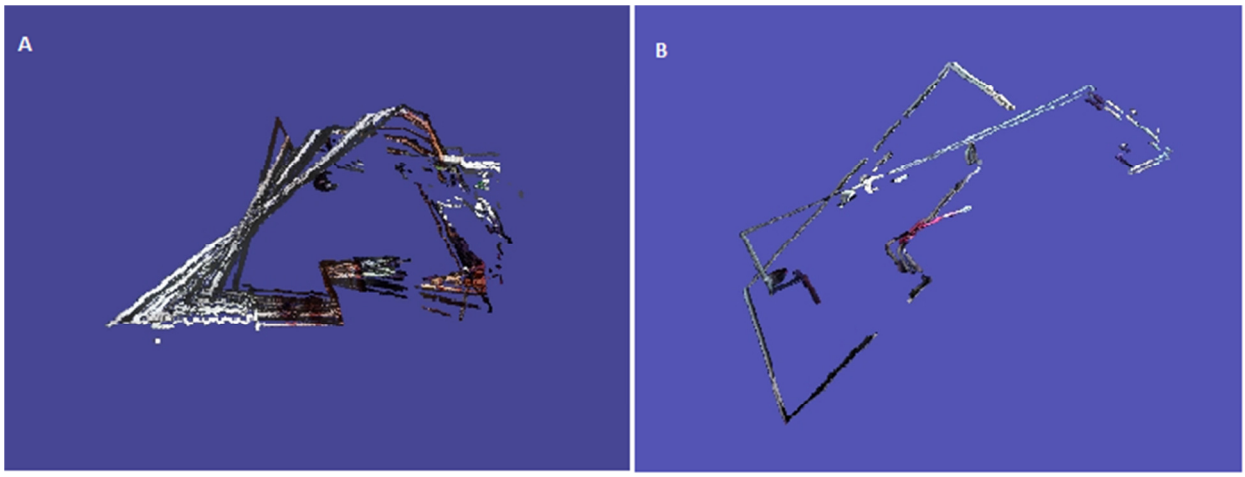


Figure 7. Images showing bad outputs from running standard RANSAC. A) First 20 extracted planes from a pointcloud containing 10.8 Million points B) First 10 extracted planes from a pointcloud containing 27.5 Million points

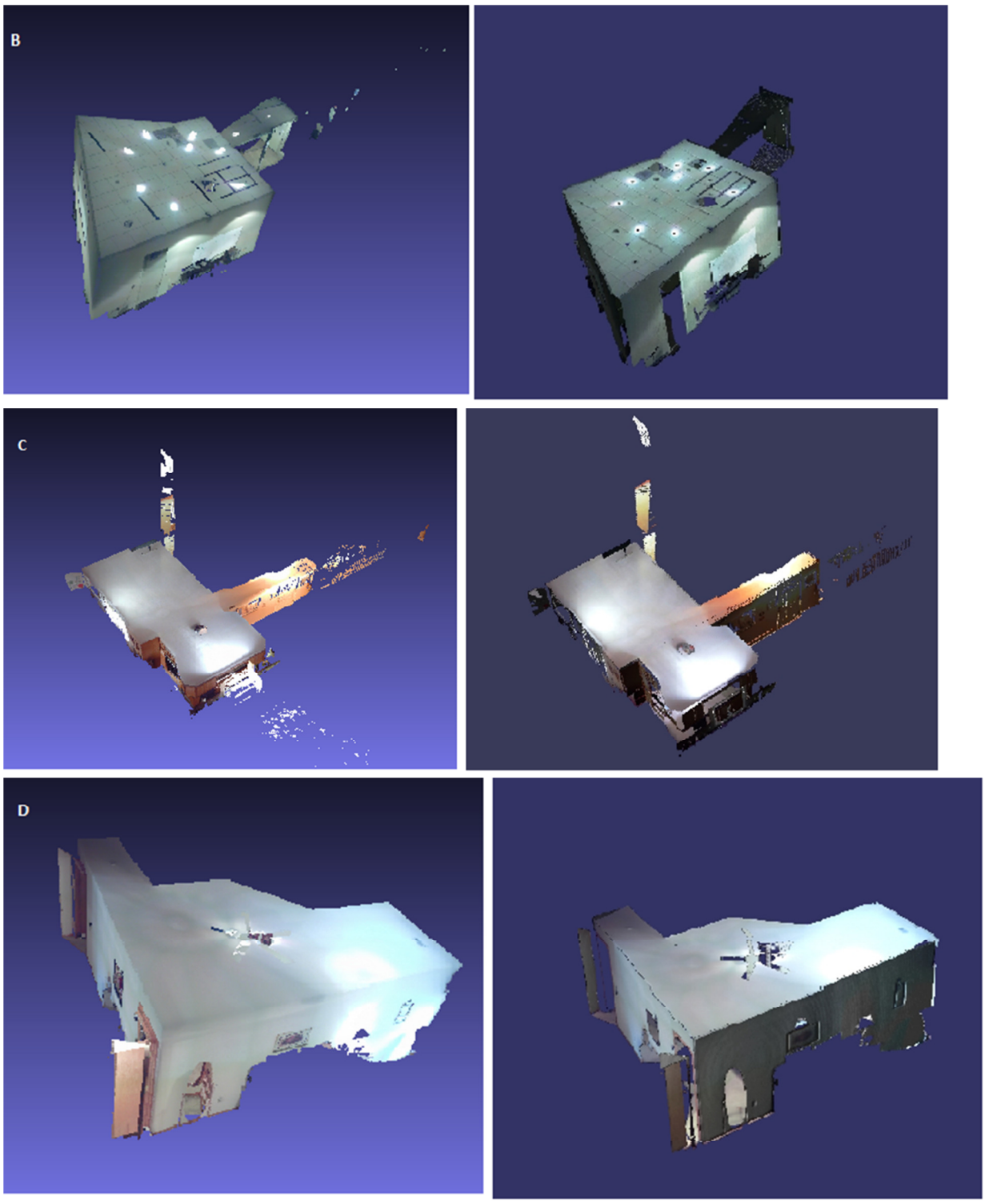


Figure 8. Figures showing comparison between original pointclouds and planar models extracted using the HPE method. A) pointcloud with 10.0 Million points B) pointcloud with 10.0 Million points C) pointcloud with 10.8 Million points D) pointcloud with 27.5 Million points. Note that the models are rendered with artificial lighting and shaders.

ORIGINAL FILE		PLANAR MODEL		POINTS LEFT	
SIZE	**POINTS**	**SIZE**	**POINTS**	**SIZE**	**POINTS**
155 MB *(figure 8a)*	10.0 Million	0.5 MB	7.3 Million	40 MB	2.6 Million
156 MB *(figure 8b)*	10.0 Million	6.5 MB	7.3 Million	22.5 MB	2.7 Million
168 MB *(figure 8c)*	10.7 Million	3.5 MB	8.7 Million	44 MB	2.1 Million
320 MB *(figure 8d)*	27.5 Million	3 MB	18.6 Million	140 MB	8.9 Million

Table 2. Results showing the reduction of four similarly sized in pointcloud datasets after utilizing the HPE method (datasets same as in Table1). The method was able to successfully abstract the vast majority of the datasets into textured planes creating file sizes that were a fraction of the original datasets

It is to be noted that the time taken to extract planes from the pointcloud does not depend directly upon the number of points in the data but predominantly on the orientation of the data model. Inputs with more planes are found to run faster than pointclouds with fewer planar surfaces.

It is also observed that the plane extraction using standard RANSAC obtained erroneous results on big data sets. The HPE method obtained consistent results irrespective of the size of the input data set. The planes looked well-defined, bounded and aligned with each other. The size of the original file was substantially reduced after obtaining the planar models.

Discussion

It is evident from the results that the HPE method would efficiently extract planes irrespective of the size and orientation of the pointcloud and even where the standard RANSAC method would generally fail. The only parameters that the HPE method would need the user to specify are the distance threshold of the plane and the number of planes to be extracted. The average points per hash, the hash dimension and the number of hash tables are dynamically calculated.

We note that for one tested instance the standard RANSAC was faster than the HPE method as shown in Table 1. For small and simple datasets (e.g. a room in which most of the points correspond to a small number of walls), the probability of the standard RANSAC method finding these planes is very high. In this special case, the hierarchical approach described in this paper is not essential for finding planes. However, while the HPE method is slower for this condition, the resulting determined planes are identical. Future work will aim to create a method to determine when the HPE method can be short-circuited by a standard RANSAC approach.

The planes found were saved as tiles of PNGs. This helps to increase the quality of the image and also does not waste space by saving pixels that do not have RGB value ($\alpha=0$). The

texture size of the planes that are saved in the PNG format could also be calculated at run-time from the point density of the plane found.

This tiling method also enabled holes in the plain, such as doorways and windows, to be maintained. In order to reduce false positives, we created minimum area and density requirements that needed to be satisfied for a plane to be retained. These user defined parameters also enabled the granularity of the resulting model to be defined.

Conclusion

Plane extraction on big data sets containing more than a million points using conventional methods like RANSAC is often inaccurate and time consuming. We propose the HPE method in which the big data sets are broken into small data sets and planes are extracted in many levels until we obtain a set of candidate points representing the original point cloud. Running RANSAC on the set of candidate points is faster and accurate which is used to obtain textured planes by referencing the original pointcloud.

The HPE method gives rise to the possibility of developing a hybrid viewer of points and textures which will effectively reduce the size of the data in memory while also preserving the quality of the pointcloud. Other image enhancing algorithms such as texture synthesis and hole filling can be used to enhance the quality of the textures output from the HPE method.

Acknowledgements

The authors would like to thank Ross Tredinnick, Gail Casper, and Patricia F. Brennan for their support on this project. This project was supported by grant number R01HS022548 from the Agency for Healthcare Research and Quality. The content is solely the responsibility of the authors and does not necessarily represent the official views of the Agency for Healthcare Research and Quality.

References

1. Chen, Baoquan, and Minh Xuan Nguyen. "POP: A hybrid point and polygon rendering system for large data." In *Proceedings of the conference on Visualization'01*, pp. 45-52. IEEE Computer Society, 2001.
2. Chum, Ondrej, and Jırı Matas. "Randomized RANSAC with Td, d test." In *Proc. British Machine Vision Conference*, vol. 2, pp. 448-457. 2002.
3. Clarke, John C., S. Carlsson, and Andrew Zisserman. "Detecting and Tracking Linear Features Efficiently." In *BMVC*, pp. 1-10. 1996.
4. Elseberg, Jan, Dorit Borrmann, and Andreas Nuchter. "Efficient processing of large 3d point clouds." In *Information, Communication and Automation Technologies (ICAT), 2011 XXIII International Symposium on*, pp. 1-7. IEEE, 2011.
5. Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM* 24, no. 6 (1981): 381-395.
6. Frisken, Sarah F., and Ronald N. Perry. "Simple and efficient traversal methods for quadtrees and octrees." *Journal of Graphics Tools* 7, no. 3 (2002): 1-11.
7. Hoppe, Hugues, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*. Vol. 26, no. 2. ACM, 1992.

8. Hough, Paul VC. "Method and means for recognizing complex patterns." U.S. Patent 3,069,654, issued December 18, 1962.
9. Illingworth, John, and Josef Kittler. "A survey of the Hough transform." *Computer vision, graphics, and image processing* 44, no. 1 (1988): 87-116.
10. Meagher, Donald. "Geometric modeling using octree encoding." *Computer graphics and image processing* 19, no. 2 (1982): 129-147.
11. Pauly, Mark, Markus Gross, and Leif P. Kobbelt. "Efficient simplification of point-sampled surfaces." In *Proceedings of the conference on Visualization'02*, pp. 163-170. IEEE Computer Society, 2002.
12. Raguram, Rahul, Jan-Michael Frahm, and Marc Pollefeys. "A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus." In *Computer Vision–ECCV 2008*, pp. 500-513. Springer Berlin Heidelberg, 2008.
13. Remondino, Fabio, and Sabry El Hakim. "Image based 3D Modelling: A Review." *The Photogrammetric Record* 21, no. 115 (2006): 269-291.
14. Rusu, Radu Bogdan, and Steve Cousins. "3d is here: Point cloud library (pcl)." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1-4. IEEE, 2011.
15. Schnabel, Ruwen, Roland Wahl, and Reinhard Klein. "Efficient RANSAC for Point Cloud Shape Detection." In *Computer Graphics Forum*, vol. 26, no. 2, pp. 214-226. Blackwell Publishing Ltd, 2007.
16. Tarsha-Kurdi, Fayez, Tania Landes, and Pierre Grussenmeyer. "Hough-transform and extended RANSAC algorithms for automatic detection of 3d building roof planes from lidar data." *International Archives of Photogrammetry, Remote Sensing and Spatial Information Systems* 36 (2007): 407-412.
17. Teschner, Matthias, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H. Gross. "Optimized Spatial Hashing for Collision Detection of Deformable Objects." In *VMV*, vol. 3, pp. 47-54. 2003.
18. Várady, Tamás, Pál Benkő, and Géza Kós. "Reverse engineering regular objects: simple segmentation and surface fitting procedures." *International Journal of Shape Modeling* 4, no. 03n04 (1998): 127-141.
19. Vosselman, George, and Sander Dijkman. "3D building model reconstruction from point clouds and ground plans." *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences* 34, no. 3/W4 (2001): 37-44.
20. Vosselman, George, Ben GH Gorte, George Sithole, and Tahir Rabbani. "Recognising structure in laser scanner point clouds." *International archives of photogrammetry, remote sensing and spatial information sciences* 46, no. 8 (2004): 33-38.
21. Wahl, Roland, Michael Guthe, and Reinhard Klein. "Identifying planes in point-clouds for efficient hybrid rendering." In *The 13th Pacific Conference on Computer Graphics and Applications*, pp. 1-8. 2005.
22. Woo, H., E. Kang, Semyung Wang, and Kwan H. Lee. "A new segmentation method for point cloud data." *International Journal of Machine Tools and Manufacture* 42, no. 2 (2002): 167-178.
23. Wu Leif Kobbelt, Jianhua. "Structure recovery via hybrid variational surface approximation." In *Computer Graphics Forum*, vol. 24, no. 3, pp. 277-284. Blackwell Publishing, Inc, 2005.

Author Bio:
Naveen Anand Subramaniam is a Master's student at the Department of Electrical and Computer Engineering in the University of Wisconsin-Madison. Although he is majoring in Computer Architecture, computer graphics and algorithms have been a parallel interest since his undergrad. With the advent of the vizHOME project in the Living Environments Laboratory, University of Wisconsin-Madison, Naveen Anand has been working with LiDAR and 3D pointclouds thus leading to the development of this paper.

Kevin Ponto:
Kevin Ponto is an Assistant Professor in the Department of Design Studies in the School of Human Ecology and in the Living Environments Laboratory at the Wisconsin Institute of Discovery at the University of Wisconsin-Madison. Kevin had previously worked with 3D scanning for cultural heritage projects, but as an investigator on the vizHOME project has turned his focus to the reconstruction of home environments to be experienced virtual reality.