

Streamlining Point Cloud Post-Processing Using Principal Component Variance, Distribution Evaluation, and Other Statistical Metrics

Michael Holm, Jack Miller, Adam Kohl, Eliot Winer
Iowa State University
Ames, Iowa

mdholm@iastate.edu, jackm@iastate.edu, adamkohl@iastate.edu, ewiner@iastate.edu

ABSTRACT

The 3D scanning market is predicted to rise by 10.2% annually through 2026 to a size of nearly 11 billion dollars (*3D Scanning Market to Rise at 10.2% CAGR till 2026*, 2021). Additionally, in 2018, it was reported that the U.S. Navy saved nearly \$2 million on a project by pursuing 3D scanning technologies (Eckstein, 2018). 3D scanners allow the modeling and simulation community to create digital representations of landscapes, vehicles, or other legacy objects for a variety of uses such as virtual training environments. While the use of 3D scanning is growing, problems with the technology still exist, such as erroneous data capture due to overexposure or overly dense sampling (Z. Wang & Li, 2020). These issues result in point clouds that are unwieldy and challenging to use. Current post-processing techniques for point clouds often require a “guess and check” method of determining proper parameters for cleaning unwanted points or reducing (i.e., downsampling) the number of points in the cloud. This takes a high number of iterations, and significant time to produce a usable point cloud model.

This paper presents a computational study with the purpose of reducing point cloud noise and generating a clean model quickly and efficiently. Using open-source libraries, the potential of mathematically determining robust parameters for operations performed on point clouds, such as noise filters, was investigated. The outcomes of this research were the relationships between common point cloud post-processing operations and standard point cloud metrics, such as principal component variance and total file size. These relationships help to identify parameter input values for a post-processing workflow that produces reliable point clouds with little to no iteration. The resulting workflow was applied to several models and produced accurate, effective, and consistent results, along with smaller file sizes. Analysis of the results also showed potential for automating this process in future work to further ease these post-processing activities.

Keywords: Point Cloud, 3D Scanning, Noise Filtration, Python

ABOUT THE AUTHORS

Michael Holm is a Master’s student in Mechanical Engineering at Iowa State University. He is currently researching how point clouds and 3D scanning can be used to create a true digital twin of a manufacturing facility. His other research interests are machine learning and simulation.

Jack Miller is a Ph.D. candidate in Computer Engineering and Human-Computer Interaction at the Iowa State University’s Virtual Reality Applications Center. His current research interests include exploring multi-user experiences in virtual and augmented reality environments.

Adam Kohl is a Ph.D. candidate in Mechanical Engineering and Computer Engineering at Iowa State University's Virtual Reality Applications Center. His research interests include the development of pattern recognition techniques and supervised learning methods for engineering applications.

Eliot Winer, P.hD., is the director of the Virtual Reality Applications Center and professor of Mechanical Engineering, Electrical and Computer Engineering, and Aerospace Engineering at Iowa State University. Dr. Winer has over 20 years of experience working in virtual reality and 3D computer graphics technologies on sponsored projects for the Department of Defense, Air Force Office of Scientific Research, Department of the Army, National Science Foundation, Department of Agriculture, Boeing, John Deere, and the Federal Highway Administration.

Streamlining Point Cloud Post-Processing Using Principal Component Variance, Distribution Evaluation, and Other Statistical Metrics

Michael Holm, Jack Miller, Adam Kohl, Eliot Winer
Iowa State University
Ames, IA

mdholm@iastate.edu, jackm@iastate.edu, adamkohl@iastate.edu, ewiner@iastate.edu

INTRODUCTION

In America alone, the 3D scanning market represents a size of \$1.23 billion dollars. Additionally, this market is expected to grow at an annualized rate of 10.26% through 2026 (Eckstein, 2018). The large and incredibly fast-growing state of this industry is representative of the vast array of uses and applications for 3D scanning technologies. From reverse engineering competing products (Helle & Lemu, 2021) to digitizing structures that do not have CAD models (Jain et al., 2011) to tracking real-world objects using computer vision techniques (Noonan et al., 2011), 3D scanning has been applied in many modern STEM fields.

The US military has a vested interest in developing these technologies as well. In 2018, the U.S. Navy saved nearly 2 million dollars on a single project by pursuing 3D scanning technologies during the planning effort for the USS *George Washington*'s refueling and overhaul. A small team of engineers were able to use a LiDAR system to accomplish the work of the usual 20-person team. (Eckstein, 2018). Additionally, in 2021, the U.S. Air Force officially issued a national call for the development of 3D scanning technology to be applied to its aging fleet, using the technology to scan and then 3D print parts that are otherwise difficult or impossible to obtain. (Goulding, 2021).

While 3D scanning is an extremely beneficial technology in the modern world, there are several issues that still exist in the field including noisy data collection, resulting from overexposure and reflective surfaces (Y. Wang & Feng, 2016), and overly dense sampling (Asy'ari Arief et al., 2019). These issues create point clouds that are larger than necessary, and difficult to interpret or use for their intended purpose.

To address these issues, several open-source application programming interfaces (APIs) have been developed from various research projects such as MeshLab (Cignoni et al., 2008) and Open3D (Zhou et al., 2018b). These APIs include various algorithms, or filters, to "clean up" noisy and dense point clouds. These filters work well when used with appropriate input parameters but determining these values can be time and resource consuming, requiring many iterations in a "guess-and-check" approach. This paper presents a computational study to examine input parameter values and provide more systematic method of choosing them.

The remainder of this paper is organized in the following order: First, the background of this field will be reviewed, giving a short overview of its history, and evolving applications. This includes a literature review discussing previous research, various libraries made for post-processing these clouds and the performance of these libraries. Following this, a methodology section will present how the above questions were investigated. Finally, several test cases will be presented, followed by results and conclusions.

BACKGROUND

Prior 3D scanner development

3D scanning technology was first developed in the 1960s as a response to rapidly model irregular physical surfaces in a virtual environment. Early scanners used lights, cameras, and projectors resulting in scans that took an unacceptable amount of time and effort to accurately create virtual surfaces. The last two decades have seen unprecedented advances in 3D scanning technologies to increase accuracy, speed, and affordability of the technology (Edl et al., 2018).

One of these advances, known as Light Detection and Ranging (LiDAR), works by emitting a laser and timing how long it takes to return to the scanner (Raj et al., 2020). In 3D LiDAR scanners, multiple lasers are used, from various angles, to record data of a 3D surface. The data collected using these scanners needed to be recorded in some compact, non-proprietary, file type, giving rise to a need for a standardized file type for the industry.

In 2003, the LAS (LASer) file format was developed to record the 3D data being collected by these new laser 3D scanners, known as point clouds (Library of Congress, 2019a). Further file types were developed for these 3D scanners, improving upon the LASer filetype, resulting in the development of the common PCD and PLY file types. The PCD file type, created for the Point Cloud Library (Rusu, 2020), provided access to a standard point cloud filetype to be used with C++, while the PLY filetype allowed additional data to be stored, like color, transparency, and surface normal (Library of Congress, 2019b).

Open-source point cloud libraries

Once a point cloud is created by a scanning system, there is still much work to be done. Raw point clouds need processing in noise reduction (Han et al., 2017), file size reduction (Suchocki & Błaszczak-Bąk, 2019), and creating usable 3D mesh files (Zwicker & Gotsman, 2004). Many teams have published various approaches to these efforts of denoising, file reduction, and mesh creation, but efforts have continued in these areas because these algorithms can always be made faster, more efficient, or more precise. One group that made a concerted effort to present open-source point cloud post-processing techniques is the group behind the Point Cloud Library, commonly known as the PCL (Rusu & Cousins, 2011). This open-source library was created in 2011 with the intention of making an open-source library available to the international community. This library, written in C++, presents algorithms for filtering, feature estimation, surface reconstruction, and several other applications.

There have been several other projects created, aimed at increasing the availability of point cloud manipulation tools. These projects include MeshLab (Cignoni et al., 2008), a graphical package for processing mesh files, libigl (Panozzo & Jacobson, 2019), a C++ library with tools for discrete differential geometry manipulation, and cilantro (Zampogiannis et al., 2018), a library created for general point cloud processing.

In 2018, a team published an open-source library, known as Open3D (Zhou et al., 2018b). This library exposes a set of data structures and algorithms, in both C++ and Python, with the purpose of creating a fast, light, easy to implement library of point cloud post-processing techniques. This library was specifically created as a response to the PCL's shortcomings. For this research, Open3D was used for the point cloud computations. However, the study could have been performed with any modern point cloud API.

Open3D

Open3D is an open-source library that presents a group of selected data structures and algorithms in C++ and Python. The code is maintained through an established code review process. Open3D has been used in many published research projects (Zhou et al., 2018a).

This library includes functions that can be used on many types of 3D geometry. These include point clouds, mesh files, and RGBD images. Specifically for point clouds, Open3D includes several filters created in previous research. These filters can be used to remove noise and other unnecessary data from the point cloud file. One of these filters is a function to down sample a point cloud using a voxel grid-based approach (Rusu, 2014). These down sampling and noise filters require several parameters as user input. One noise filter, called the statistical outlier removal filter, takes two arguments, called “number of neighbors” and “standard ratio”. This filter removes all points whose average distance to the nearest “number of neighbors” lies more than “standard ratio” standard deviations outside the average for the entire cloud. Similarly, a function called voxel down sampling creates uniformly distributed boxes of side length “voxel size” across the point cloud and replaces all points within each box with a single point at the average location of all points removed. These filters all require a “guess and check” approach for determining the optimal parameters for these filters. This paper investigated the potential of graphing relationships between these filters and the optimal parameters, to find a way to determine the best filter parameters possible more quickly.

METHODOLOGY

Voxel Down Sample Graphing

The first filter investigated in this project was the voxel down sample filter described earlier. To accomplish this, a function was created to graph the voxel down sample filter “voxel size” parameter against the total file size in bytes of the output cloud, and the average distance between each point and its nearest neighbor. An example of these two graphs can be seen in Figure 1. Various point clouds were

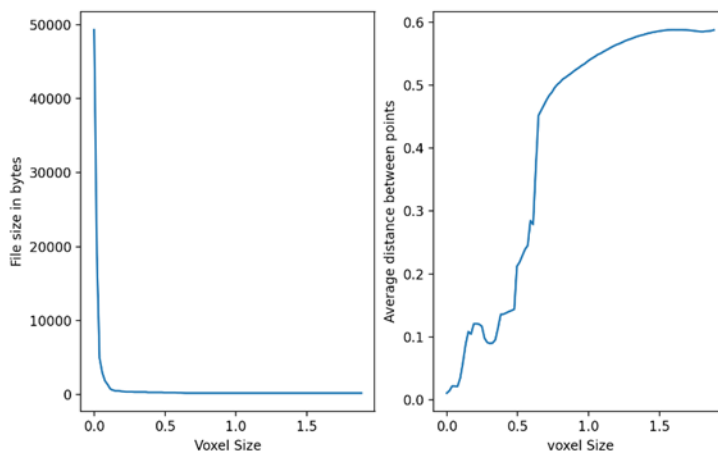


Figure 1: An example of the graph created for the voxel down sample testing

tested with this approach, and the resulting graphs were compiled to try to discover a pattern in the overall shapes of these graphs created.

Statistical outlier removal graphing

An additional function was created to graph the statistical outlier removal function’s parameters against the variance along the principal component axes of the input point cloud. The principal component axes were found by using scikit-learn, a common machine learning and data analysis module for Python (Pedegrosa et al, 2011). Using the principal component

analysis function in this module, the three principal component axes were found. After finding the axes, the axis that accounted for the smallest percentage of variance in the data set was isolated. This percentage, called the principal component least variance ratio was graphed against the two parameters of the statistical outlier removal

function, using Matplotlib's 3D plot feature. An example of the graph created in this manner can be seen in Figure 2.

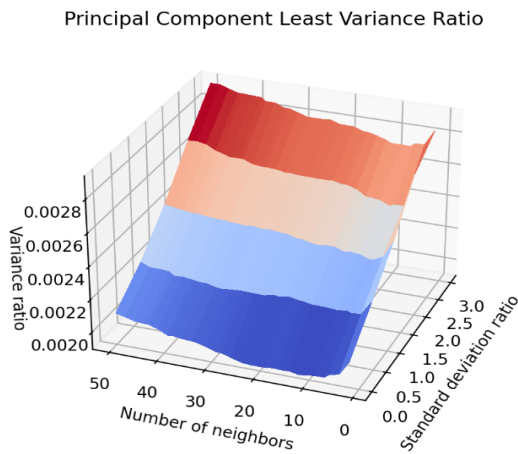


Figure 2: Graph created in the statistical outlier removal filter graphing

community to use for various applications, including point cloud completion, point cloud registration, and other standard industry processes. Using this database, ten point cloud files were selected to test these filters, and results were then evaluated. An example of a point cloud taken from this database is shown in Figure 4.

Noise Generation



Figure 4: A guitar point cloud from the MVP database

One area in which the field of point cloud denoising research is lacking is a standard database of noisy point clouds. To address this, point clouds taken from the MVP database were run through a function to generate artificial noise. This function followed a process very similar to other research teams (Rakotosaona et al., 2020) by taking a mean argument, a standard deviation multiplier argument, and a noise percent argument to produce errors in a point cloud. This function then randomly selected a percentage of points in the input point cloud and moved these points in a random direction. The magnitude of this movement was decided by using a normal distribution, whose average was the mean argument, and whose standard deviation was the average distance between each point and its closest point in the cloud, multiplied by the standard deviation multiplier argument. In this paper, values of 0, 2, and .2, were used for the mean, standard deviation multiplication, and noise percentage arguments, respectively. These parameters created a suitably noisy point cloud, while still preserving the base shape of the

Finally, a function was created to graph the statistical outlier removal filter parameters against the average distance between each point, and its closest point in the graph. This graph was then plotted using Matplotlib's 3D graph function (Hunter, 2007), similar to the statistical outlier removal graphing function. An example of a graph created in this function can be seen in Figure 3, below.

MVP Point Cloud Database

To provide a possible benchmark to compare this research against past publications, and against future efforts, the Multi-View Partial (MVP) Point Cloud Database was used to evaluate these filters and perform this parameter study (Pan et al., 2021). This database was created to provide a benchmark for the point cloud

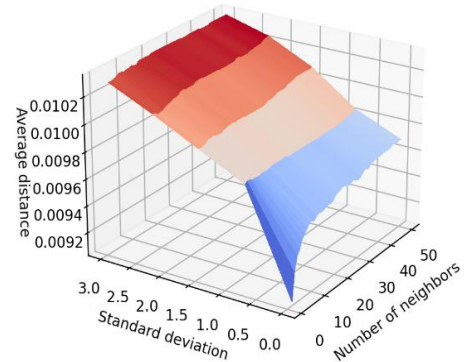


Figure 3: Graph created in the average distance graphing

unedited cloud. Other values were tested and produced similar results as well. An image of a noisy point cloud, generated through this process, can be seen in Figure 5.

Experimentation

The actual process for experimentation using these functions described above is as follows: First, a clean point cloud was taken from the MVP Point Cloud Database. Second, noise was then generated using the function described earlier. Third, the noisy point cloud was then run through the statistical outlier removal graphing function. This graph was created using a number of neighbors argument ranging from one to fifty, with thirty points generated in this range, in a linear distribution. Additionally, the standard deviation axis was created using a range of .000001 to three, with five samples generated in this range, with a linear distribution. These values were selected for this function because it was found that the only relevant location in these graphs created was a global minimum. No other location on the surface created provided consistent filter results, and there were no other recognizable graphical features created, e.g., local maxima, inflection points, and other local minima.

When testing larger ranges, no new minimum locations were produced in these graphs, but calculation time was dramatically increased.

Additionally, it was found that increasing the resolution of these graphs provided no valuable increase in the accuracy of the global minimum. For example, when doubling the resolution of this graph, it was found that the minimum value found changed approximately .06%, on average. This information was then graphed against both the average distance between a point, its nearest point, and the principal component least variance ratio. The same process was repeated using the voxel down sample graphing function, with a voxel size axis resolution of 100 points between .0001 to 2. The selection of this range and resolution followed a similar logic as the previous function described, showing a diminishing effect at larger voxel sizes, resulting in a plateau at values past 2, and a completely empty point cloud file at values near 2.5. Again, increasing resolution provided no valuable increase in accuracy of the data collected, but did increase computational time.

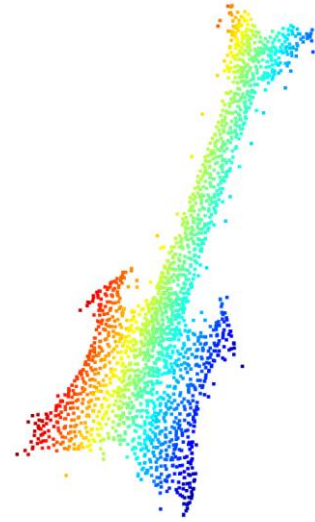


Figure 5: A noisy point cloud generated from the MVP database

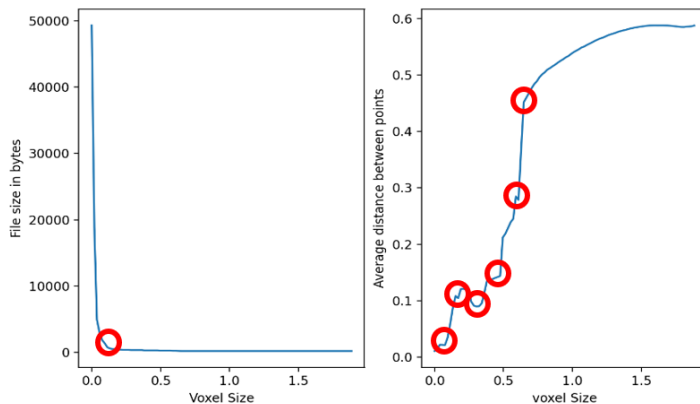


Figure 6: Locations of interest marked on the voxel down sample graph

The process described above was repeated for ten point clouds in the MVP database. These point clouds included a sofa, a chair, a guitar, a lamp, and other everyday objects. After creating these graphs, several “locations of interest” were selected in these graphs. These locations of interest were local maxima, minima, inflection points, and sudden changes in curvature. Mathematically these are points in the graphs where derivative values change dramatically. The parameter values corresponding to these points were used as inputs for

the various filters. After using these parameters in these filters the results were compared to the clean, unedited point clouds. Figure 6 shows the described graphs with the all locations of interest circled in red.

Upon comparison of the results of these various filter parameters, several patterns were found between the graphs of these filters and the parameter values discussed in the results section of this paper. The goal is to create a filtered point cloud that is significantly reduced in size, but can create a suitable mesh model.

RESULTS AND DISCUSSION

Voxel Down Sample Graphing

By following the processes described above for the voxel down sampling graphing function, several patterns were found in the graphs created. Figure 7 shows graphs created, a specific location of interest (circled in red), the original point cloud, and a filtered point cloud created using parameters from the location of interest.

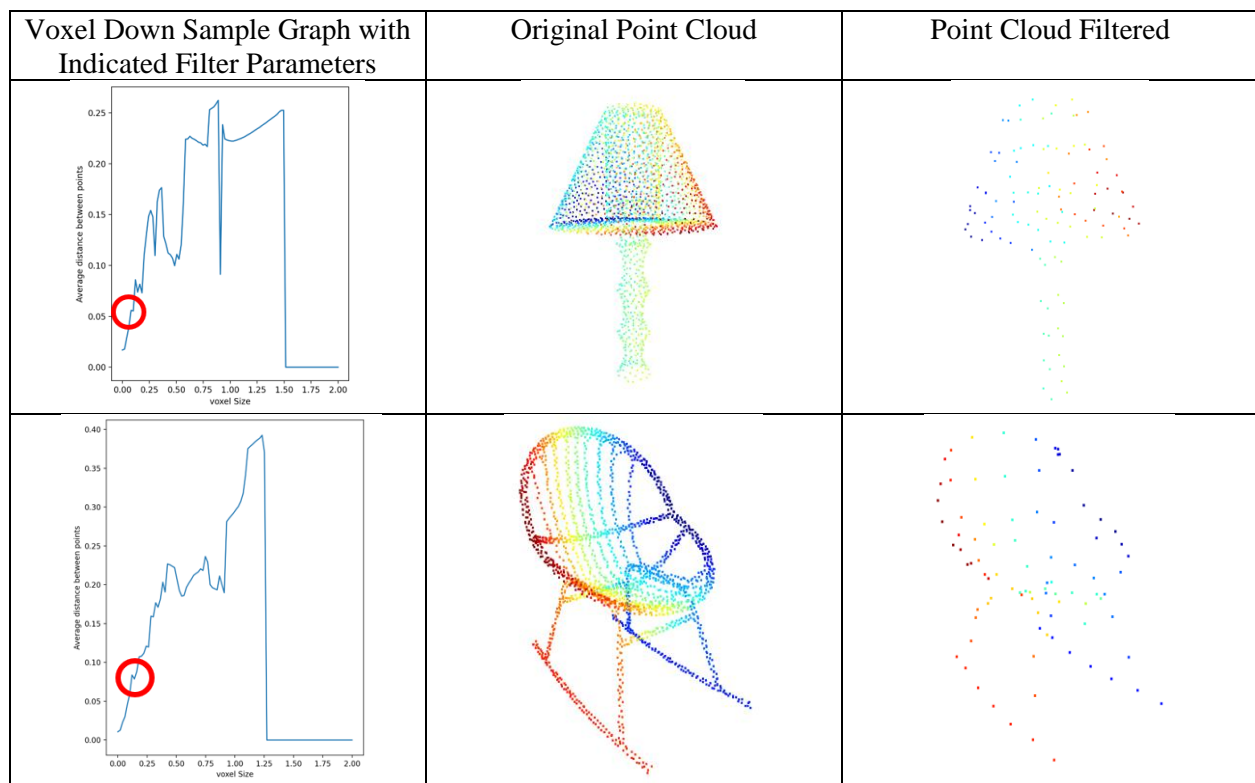


Figure 7: Table showing results of voxel down sample graphing

As stated earlier, ten point clouds were tested using this approach, with all locations of interest identified and examined. It was found that when graphing average distance between a point, and its nearest point, as a function of voxel size the function will behave smoothly for small voxel size values. However, there will be a critical point where the function begins to exhibit erratic behavior, seen with the first sharp change in its derivative, as shown in Figure 7. This change in the function behavior represents the boundary for what could be called “acceptable” resolution in the point cloud. Using this point, a user can gauge the upper limits of what points can be removed while still leaving a recognizable shape. If this relationship holds true, a point cloud could be reduced significantly in size, while assuring a user that the resulting filtered cloud can be meshed for a usable virtual model. Point clouds could be processed once, with minimal time and computational resources, instead of undergoing multiple iterations in a standard

“guess and check” approach. For example, setting the voxel down sample parameter to half of the critical point value produced a point cloud that is consistently reduced in size substantially, while leaving enough data to create a very low poly mesh. An example is shown below, in Figure 8.


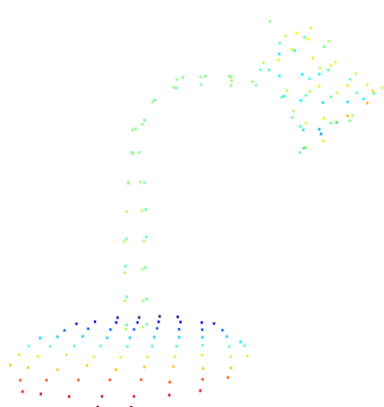

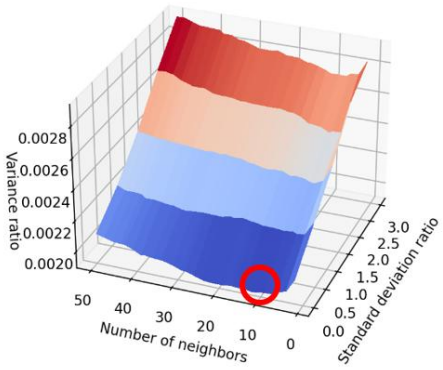
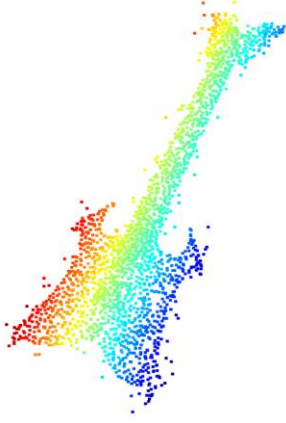
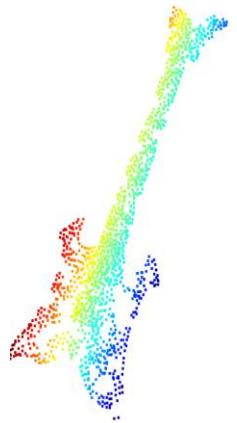
Unfiltered Point Cloud	Filter Parameters at one half of the Critical Point	Mesh Created
		

Figure 8: Example of filter parameters at one half the critical point

Statistical Outlier Removal Graphing

The statistical outlier removal graph was then investigated, with additional patterns identified. Figures 9a – 9d show four such graphs created (with locations of interest circled in red), the noisy MVP point clouds and the corresponding filtered point cloud. Figures 9a and 9b show results from the principal component least variance ratio graphing. Figures 9c and 9d show results from the average distance between points graphing. Additionally, the areas circled in red on the graphs shown indicate the statistical outlier removal parameters used for the filtered point cloud shown in the far-right column.

Label	Statistical Outlier Removal Graph with Indicated Filter Parameters	Noisy Point Cloud	Point Cloud Filtered
9a	<p>Principal Component Least Variance Ratio</p> 		

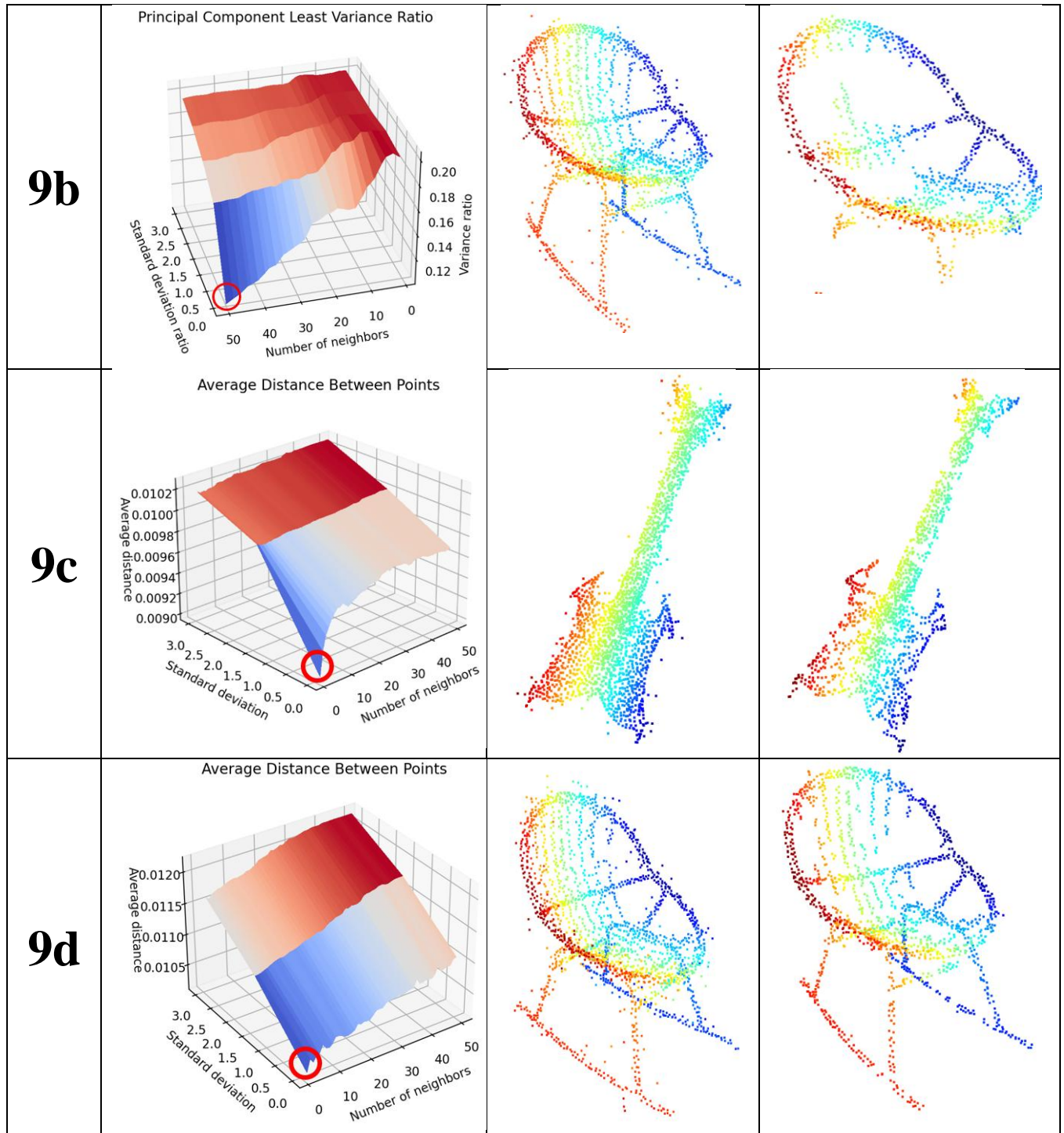


Figure 9a: Results when applying least variance ratio approach to a guitar MVP point cloud

Figure 9b: Results when applying least variance ratio approach to a chair MVP point cloud

Figure 9c: Results when applying average distance approach to a guitar MVP point cloud

Figure 9d: Results when applying average distance approach to a chair MVP point cloud

The graphs shown above are examples of the patterns found when graphing the parameters of Open3D's statistical outlier removal function against both the average distance between points, and the principal

component least variance ratio. It was found that for both these approaches, the global minimum of the graph created gives consistent, satisfactory results. That is, a point cloud that is reduced in size, with little to no noise, that can still be meshed. However, as can be seen in the first and second row of the table above, when using the principal component least variance ratio approach, satisfactory results are only produced when the clean point cloud has a predominantly planar shape to it.

Evaluation

The patterns proposed in this paper were quantitatively evaluated based on consistency, accuracy, and effectiveness. The voxel down sampling pattern proposed, using the critical point as a reference for a cap on the voxel size, was evaluated by following this pattern on the ten point clouds selected from the MVP database, and recording file size reduction results. This file size reduction was calculated as the percentage change in number of total points in the point cloud. When following this method of evaluating, an average file size reduction of 96.68% was found, with a standard deviation of 2.13%. It is important to remember however, that the assessment of whether this is an effective pattern or not cannot be measured by the average reduction alone, but rather the consistency of the results as well as visually if a usable model can be meshed.

Additionally, the two patterns discussed for the statistical outlier removal filter were evaluated in a similar method. This evaluation was done by following the patterns shown in Figure 12 for all ten point clouds selected from the MVP database. After filtering each point cloud, several data points were gathered, including total points removed by the filter applied, and how many of the points were truly noisy points. Additionally, these numbers were compared to the original point cloud size and the original number of noisy points. Upon evaluation of both patterns discussed, it was found that when using the principal component least variance ratio approach, an average of 43.8% of the points removed were correctly identified as noise, with a standard deviation of 20%. Additionally, it was found that on average, 56% of the total noise was removed with this approach, with a standard deviation of 15%.

Similarly, when using the average distance between points approach an average of 32.8% of the points removed were correctly identified as noise, with a standard deviation of 11%. Additionally, it was found that on average, 53% of the total noise was removed with this approach, with a standard deviation of 7%.

Discussion

Ultimately, these results indicate several important points of information regarding the investigation of these patterns. First, using the critical point approach for the voxel down sampling filter, it was found that the results produced are consistent, with a standard deviation of 2.13% in total file size reduction. This means that this approach could be uniformly applied to collected point clouds and would result in predictable results in both file size and file quality. This ability to standardize the approach of voxel down sampling would achieve the stated goal of increasing the speed of applying these filters, while achieving satisfactory results.

Additionally, the results gathered from the statistical outlier removal patterns discussed provide similar results. Overall, the standard deviation of the results shown are low enough to produce predictable results for point clouds filtered. While the accuracy and effectiveness of this patterns may leave something to be desired, the predictable quality of these patterns mean that, if these accuracy and effectiveness metrics are satisfactory for the intended application of these filters, both approaches would work wonderfully in achieving the goal of increasing point cloud post processing speed.

FUTURE WORK

The findings presented in this paper present many areas of future work. The first of these areas is further evaluation of the critical point for the voxel down sampling filter. For some point clouds, it can be difficult to determine where this critical point truly is. This point could potentially be further identified by using information of the derivative and the second derivative of the average distance between points with respect to the voxel size used in this filter. This derivative information could be used to further define where a critical point for a point cloud truly is. Similarly, further investigation into patterns for Open3D's statistical outlier removal function could lead to the discovery of approaches that produce more consistent, accurate, and effective results than the approaches presented in this paper.

There is also potential for the automation of these patterns. For example, using the principal component least variance ratio approach, a workflow could be created that utilizes gradient descent techniques to automatically determine the location of a local minimum in this graph, whose location could then be used as the parameters of Open3D's statistical outlier removal function. A gradient descent technique could also be used for the average distance between points approach. Automating these patterns would lead to further time saving in the post processing workflow of points clouds.

Finally, a time study could be done on the speed of these approaches. The graphs used in all three approaches take a non-negligible amount of time to calculate. A time study could be performed to evaluate the effectiveness of these approaches in saving time when post-processing point clouds.

CONCLUSIONS

This paper's goal was to investigate existing relationships between statistical metrics of points clouds and the various filter parameters associated with these metrics. By investigating these relationships, it was the goal of this paper to improve the speed at which point clouds can be post processed. Specifically, the relationships between Open3D's voxel down sample function and statistical outlier removal function, and a point clouds average distance between points and principal component least variance ratio. These relationships were evaluated using the MVP point cloud database, and statistically generated noise. Several relationships were found between these filters and these point cloud metrics. These relationships were then used to filter ten point clouds from the MVP database, and evaluated on effectiveness, accuracy, and consistency in performance.

The evaluation results presented in this paper show that using these relationships provides a consistent, but potentially unsatisfactory filter performance. However, if high accuracy filter performance is not needed, and rough results are acceptable, these relationships can be easily taken advantage of to skip the "guess-and-check" methods originally required for these filters.

The relationships discussed in this paper have many possible applications in the field of post processing point clouds. Beyond the simple implementation of these graphs to quickly find suitable filter parameters, there is potential for further automation of these post-processing workflow using gradient descent techniques. Additionally, the potential of parallelizing point cloud post-processing, using the approaches discussed, presents further optimization of the point cloud post-processing workflow when using Open3D. This further workflow optimization will result in point clouds that are easier to work with and cheaper to produce, and shorter overall project timelines when using point cloud and 3D scanning technology.

REFERENCES

- 3D Scanning Market to Rise at 10.2% CAGR till 2026. (2021). GlobeNewswire. <https://www.globenewswire.com/news-release/2021/12/01/2343770/0/en/3D-Scanning-Market-to-Rise-at-10-2-CAGR-till-2026-Driven-by-Increasing-Investment-in-Product-R-D-to-Aid-Growth-and-Applications-Across-Diverse-Industry-Verticals.html>
- Asy'ari Arief, H., Arief, M. M., Bhat, M., Indahl, U. G., Tveite, H., & Zhao, D. (2019). Density-adaptive sampling for heterogeneous point cloud object segmentation in autonomous vehicle applications. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2019-June*.
- Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., & Ranzuglia, G. (2008). MeshLab: an Open-Source Mesh Processing Tool. Automatic 3D scanning View project Virtual reconstruction of the entrance of Ripoll's Monastery View project MeshLab: an Open-Source Mesh Processing Tool. *Eurographics Italian Chapter Conference*. <https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136>
- Eckstein, M. (2018). *Navy Use of Laser Scanning Already Showing Big Savings; Summit This Month to Refine Plans - USNI News*. USNI News. <https://news.usni.org/2018/07/02/navy-use-of-laser-scanning-already-showing-big-savings-summit-this-month-to-refine-plans>
- Edl, M., Mizerák, M., & Trojan, J. (2018). 3D LASER SCANNERS: HISTORY AND APPLICATIONS. *Acta Simulatio-International Scientific Journal about Simulation*, 4–5. <https://doi.org/10.22306/asim.v4i4.54>
- Goulding, C. (2021). *Air Force Issues National Call For 3D Scanner Technology* « Fabbaloo. Fabbaloo. <https://www.fabbaloo.com/news/air-force-issues-national-call-for-3d-scanner-technology>
- Han, X. F., Jin, J. S., Wang, M. J., Jiang, W., Gao, L., & Xiao, L. (2017). A review of algorithms for filtering the 3D point cloud. *Signal Processing: Image Communication*, 57, 103–112. <https://doi.org/10.1016/J.IMAGE.2017.05.009>
- Helle, R. H., & Lemu, H. G. (2021). A case study on use of 3D scanning for reverse engineering and quality control. *Materials Today: Proceedings*, 45, 5255–5262. <https://doi.org/10.1016/J.MATPR.2021.01.828>
- Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering. <https://matplotlib.org/>
- Jain, A., Soner, S., & Gadwal, A. (2011). Reverse engineering: Journey from code to design. *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*, 5, 102–106. <https://doi.org/10.1109/ICECTECH.2011.5941966>
- Library of Congress. (2019a). *LAS (LASer) File Format, Version 1.4*. Digital Preservation. <http://www.digitalpreservation.gov/formats/fdd/fdd000418.shtml>
- Library of Congress. (2019b). *Polygon File Format (PLY) Family*. Sustainability of Digital Formats: Planning for Library of Congress Collections. <https://www.loc.gov/preservation/digital/formats/fdd/fdd000501.shtml>
- Noonan, P. J., Cootes, T. F., Hallett, W. A., & Hinz, R. (2011). The design and initial calibration of an optical tracking system using the Microsoft Kinect. *IEEE Nuclear Science Symposium Conference Record*, 3614–3617. <https://doi.org/10.1109/NSSMIC.2011.6153680>

- Pan, L., Chen, X., Cai, Z., Zhang, J., Zhao, H., Yi, S., & Liu, Z. (2021). Variational Relational Point Completion Network. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. <https://doi.org/10.1109/CVPR46437.2021.00842>
- Panozzo, D., & Jacobson, A. (2019). libigl: Prototyping Geometry Processing Research in C++. *EUROGRAPHICS*. <https://doi.org/10.2312/egt.20191037>
- Pedegrosa et al. (2011). *scikit-learn Machine Learning in Python*. JMLR 12. <https://scikit-learn.org/stable/>
- Raj, T., Hashim, F. H., Huddin, A. B., Ibrahim, M. F., & Hussain, A. (2020). A Survey on LiDAR Scanning Mechanisms. *Electronics 2020, Vol. 9, Page 741, 9(5)*, 741. <https://doi.org/10.3390/ELECTRONICS9050741>
- Rakotosaona, M. J., La Barbera, V., Guerrero, P., Mitra, N. J., & Ovsjanikov, M. (2020). PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds. *Computer Graphics Forum*, 39(1), 185–203. <https://doi.org/10.1111/CGF.13753>
- Rusu, R. B. (2014). *Downsampling a PointCloud using a VoxelGrid filter. Documentation - Point Cloud Library (PCL)*. http://pointclouds.org/documentation/tutorials/voxel_grid.php#voxelgrid
- Rusu, R. B. (2020). *The PCD (Point Cloud Data) file format*. 22.8.2020. https://pointclouds.org/documentation/tutorials/pcd_file_format.html#why-a-new-file-format
- Rusu, R. B., & Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). *IEEE International Conference on Robotics and Automation*. <http://pointclouds.org>
- Suchocki, C., & Błaszczak-Bąk, W. (2019). Down-sampling of point clouds for the technical diagnostics of buildings and structures. *Geosciences (Switzerland)*, 9(2). <https://doi.org/10.3390/GEOSCIENCES9020070>
- Wang, Y., & Feng, H. Y. (2016). Effects of scanning orientation on outlier formation in 3D laser scanning of reflective surfaces. *Optics and Lasers in Engineering*, 81, 35–45. <https://doi.org/10.1016/J.OPTLASENG.2016.01.003>
- Wang, Z., & Li, B. (2020). A High Reliability 3D Scanning Measurement of the Complex Shape Rail Surface of the Electromagnetic Launcher. *Sensors 2020, Vol. 20, Page 1485, 20(5)*, 1485. <https://doi.org/10.3390/S20051485>
- Zampogiannis, K., Fermüller, C., & Aloimonos, Y. (2018). cilantro: A Lean, Versatile, and Efficient Library for Point Cloud Data Processing. *ACM International Conference on Multimedia*. <https://doi.org/10.1145/3240508.3243655>
- Zhou, Q.-Y., Park, J., & Koltun, V. (2018a). *Open3D: A Modern Library for 3D Data Processing*. <http://arxiv.org/abs/1801.09847>
- Zhou, Q.-Y., Park, J., & Koltun, V. (2018b). *Open3D: A Modern Library for 3D Data Processing*. <http://www.open3d>
- Zwicker, M., & Gotsman, C. (2004). Meshing Point Clouds Using Spherical Parameterization. *Eurographics Symposium on Point-Based Graphics*.