Two Automated Point Cloud Filtration Approaches Using Principal Component Variance and Nearest Neighbor Distance

Michael Holm, Eliot Winer

Iowa State University

Ames, Iowa

Author	Affiliation	e-mail
Michael Holm	Iowa State University	mdholm@iastate.edu
Eliot Winer	Iowa State University	ewiner@iastate.edu

ABSTRACT

Point clouds generated from 3D scans of part surfaces consist of discrete 3D points, some of which may be incorrect, or outliers. Outlier points can be caused by the scanning method, part surface attributes, and data acquisition techniques. Filtering techniques to remove these outliers from point clouds frequently require a “guess and check” method to determine proper filter parameters. This paper presents two novel approaches to automatically determine proper filter parameters using the relationships between point cloud outlier removal, principal component variance, and the average nearest neighbor distance. Two post-processing workflows were developed that reduce outlier frequency in point clouds using these relationships. These post-processing workflows were applied to point clouds with artificially generated noise and outliers, along with two real-world point clouds. Analysis of the results showed both approaches effectively reducing outlier frequency when used in suitable circumstances.

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

INTRODUCTION

In the US, 3D scanning is a $1.23 billion industry that is expected to grow at an annual rate of 10.26% through 2026 [1]. The 3D scanning industry includes various technologies, from structured light to lasers, as well as software to acquire and work with the scanned data. 3D scans are often stored as point clouds, which consist of discrete 3D points irregularly sampled from continuous surfaces [2]. Point clouds stored as .PCD and .PLY files can be used for many purposes, including mesh creation [3], precise measurement of irregular objects [4], and part inspection [5]. While substantial advancements have been made in acquiring and storing point cloud data, there are still several issues that exist with the effective utilization in many applications. One of the most consequential of these issues is the existence of outliers in a point cloud, resulting from overexposure, reflective surfaces, user error, environmental factors, and sensor malfunction [6,7].

Outlier points differ from valid measurement points or noisy points in that outliers are false measurements that do not belong to the scanned surface, have geometrical discontinuities, and sharp features [8]. Figure 1 shows a point cloud with real data shown in blue and large amounts of noise shown in red. Figure 2 shows a point cloud with many outliers shown in red. A scanned point cloud can contain both outliers and noise, caused by different sources. The difference between the two categories is that noise refers to random variations in the positional data of a scanned point, while outliers deviate significantly from the expected distribution of the scanned data [7].

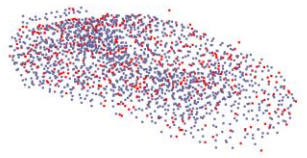

Figure 1: A point cloud with many noisy points, shown in red

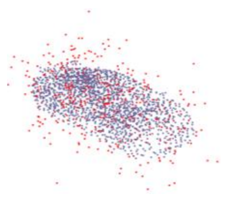

Figure 2: A point cloud with many outliers, shown in red

Outliers reduce the fidelity of point clouds and create problems during point cloud registration, meshing, and normal estimation [9,10,11]. To address the existence of outliers, and the challenges they create for point cloud applications, several open-source application programming interfaces (APIs) have been developed from various research projects such as the Point Cloud Library [12], MeshLab [13], Open3D [14], and cilantro [15]. These APIs include filters that are intended to remove outliers without affecting the scanned surface data in point clouds.

To effectively use a filter provided from one of these APIs, a user must determine appropriate input parameters so that the filtered point cloud is an accurate representation of the scanned object. However, determining these parameters is time and resource intensive. This process requires a user to guess what parameters may create a point cloud that contains fewer outliers while maintaining "good" points from the surface, and then adjusting their guess based on subjective visualization of the resulting filtered point cloud. Providing a value for the initial input parameter guess is a challenge on its own. Determining what direction to adjust parameters in subsequent iterations requires expert knowledge, intensive manual labor, and is specific for the filtration method used. Shown in Figures 3-6 is an example of the difference between appropriate and inappropriate parameters. These figures show the application of an outlier removal filter with two different sets of parameters on a point cloud that has artificially generated outliers. Figures 5 and 6 serve as a visual comparison for how ineffective a point cloud filter can become when inappropriate filtration parameters are used.

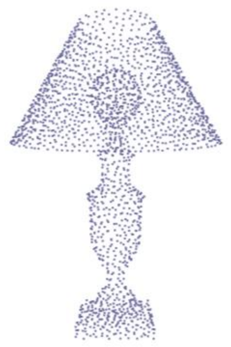
Figure 3: Original point cloud

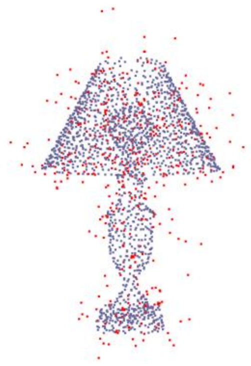
Figure 4: Original point cloud with outliers and noise shown in red

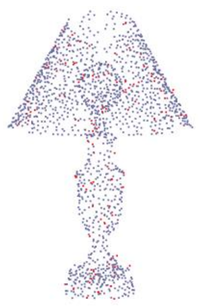
Figure 5: Filtered point cloud with appropriate parameters and noise shown in red

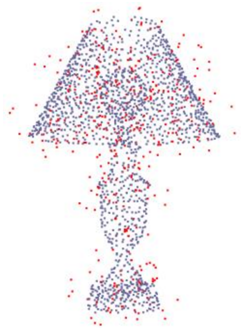
Figure 6: Filtered point cloud with inappropriate parameters, with outliers and noise shown in red

Using inappropriate parameters greatly diminishes the filter's effectiveness, leaving a point cloud with many outliers that clearly do not align with the ground truth surface. Removing noisy points typically consists of slightly adjusting points to line up with some local estimated surface [2,16]. Removing noisy points after outliers is out of the scope of the research presented in this paper.

There has been research to create outlier filters with few or zero parameters to reduce the dependency on a user's parameter choice(s) [17,18]. Wolff et al. created a filter that uses a combination of RGB information and depth maps to remove points that are most likely outliers [19]. The filter works by analyzing the color and location of a point. If the point's location is inconsistent with a collection of separately collected depth maps, or if its color is inconsistent with its nearest neighbors in the point cloud, it is removed. This filter requires a total of only two user defined parameters, which are both used to define thresholds for how different a point must be from its recorded depth or nearby colors to be removed. Wolff's filter performs well, because it removes more noise and outliers and maintains more ground truth detail when compared to other filters. However, it requires both RGB information and depth map information for the filtered point cloud, which is not always available to a user.

ML methods have also been used to filter point clouds contaminated with both outliers and noise. PointCleanNet is an ML approach, not requiring any user parameters, that deletes any outliers from the point cloud and corrects noisy points by moving them an estimated distance along an estimated vector so that they line up with a locally fit ground truth plane [20]. PointCleanNet was shown to perform well by performing quantitative analysis on its performance when tested with artificially generated noise and outliers. It was also tested on a variety of real-world point clouds, visually comparing the method's results to established filtration methods. PointCleanNet requires a large training dataset, consisting of over 500 noisy points clouds paired with ground truth point clouds. Generating a training dataset of this size, and training PointCleanNet using this dataset, requires expert knowledge that a standard user may not have. In addition, PointCleanNet's black box nature means that troubleshooting any lack in performance is onerous and requires intimate knowledge of PointCleanNet's architecutre.

As 3D scanning becomes ever more prevalent in industry, the wide range of new users and use cases continues to expand the 3D scanning industry. Newly developed APIs provide tools to filter point clouds, but they do not address issues related to a user's ability to select the appropriate filter tool to remove outliers. It is important to address this issue because outliers cause problems with point cloud registration, object recognition, and meshing. Many outlier filtration methods have been created in previous research, traditionally requiring input parameters provided by an expert user. The performance of these filters is heavily dependent on the selection of suitable parameters. Modern research has worked to reduce or eliminate the need for user provided parameters, seen in Wolff's filter and PointCleanNet. However, these modern filtration processes all leave something to be desired in their effectiveness. This has created a need for an outlier filtration process that doesn't require parameters provided by an expert user, is applicable to a wide range of point cloud geometries and outlier distributions, and is easy to troubleshoot when it fails to perform satisfactorily.

BACKGROUND

3D scanner development

3D scanning technology was first developed in the 1960s as a technique to rapidly create 3D models of irregular physical surfaces using lights, cameras, and projectors. The first 3D scanning systems took a long time to generate a representation of the physical surface that was not very accurate [21]. The large time and monetary costs meant that 3D scanning was primarily funded through government programs, not industry customers [22]. However, government funded research work resulted in important advances in the field of 3D scanning, including an early time of flight (ToF) 3D scanner created by NASA's Jet Propulsion Lab in 1977 [23]. The ToF scanner emitted laser pulses to reflect back to the scanner, which were then recorded to reconstruct the scanned object(s).

In the last two decades there have been unprecedented advances in 3D scanning technologies that increased accuracy, speed, and affordability of the technology [21]. One of these advances, known as Light Detection and Ranging (LiDAR), works by emitting multiple laser pulses and timing how long they take to return to the scanner [24]. LiDAR scanners use multiple lasers at various angles, to record 3D surface data. LiDAR scanning has been used in a wide array of applications, including capturing geographical information quickly for surveyors [25], measuring the movement of gas in a system [26], and for non-invasively analyzing delicate anthropological artifacts [27]. Parallel to this development of new scanning technology, problems with outliers began to be discussed in research [3].

As LiDAR scanners became widely used, the data collected needed to be recorded in a memory-efficient, non-proprietary industry standard filetype. The LAS (LASer) file format was developed to record the data being collected by LiDAR 3D scanners, referred to as point clouds [28]. Other filetypes were developed for these 3D scanners, improving upon the LASer filetype. The PCD filetype, created for the Point Cloud

Library [29], 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 normals [30]. The PLY filetype enabled development of many new approaches for filtering a point cloud, by utilizing a point's color, normal, and transparency, compared to its neighbors.

Post-processing Software Advances

Post-processing to improve the quality of the point cloud representation of the scanned object takes expertise and can be very time consuming. Raw point clouds (i.e., point clouds that haven't been post-processed) often require a combination of noise reduction [31], file size reduction [32], and 3D mesh creation [33]. Published approaches for filtering [19,18], file reduction [32,34], and 3D mesh creation [35,3] were found to be best suited for specific applications. For example, statistical based approaches to outlier filtering are best suited for uniformly distributed outliers [36], while projection-based methods are best suited for point clouds with generally flat surfaces [37]. The wide array of post-processing techniques has resulted in a diverse set of point cloud manipulation tools. Expertise is required to select the appropriate point cloud manipulation tool. Selection of the appropriate tool improves the quality of the post-processed point-cloud but takes significant time.

To help users choose appropriate point cloud post-processing techniques for their part(s), research groups have focused on making it easier to choose from the wide array of point cloud manipulation tools. The Point Cloud Library, commonly known as PCL, [12] has made concerted efforts to help users choose open-source point cloud post-processing techniques. This open-source library was created in 2011 with the intention of making a toolbox easily available to the international community. PCL contains algorithms for filtering, feature estimation, surface reconstruction, and several other tools that help users post-process point clouds.

An example of one of these algorithms is the moving least squares upsampling algorithm. Methods used to upsample a point cloud (i.e., increase the number of points in the point cloud while maintaining the overall geometry) [38,39] aid in point cloud post-processing by restoring detailed geometric information[40] and improving the performance of point cloud object detection applications[41]. PCL's moving least squares algorithm provides a method to fit a surface to a neighborhood of points [42]. Using this algorithm, a user can increase the density of points in a point cloud, while maintaining the overall shape and geometric features of that point cloud. Figure 7 shows the original point cloud for a chair, and Figure 8 shows an example of a point cloud that has been upsampled.

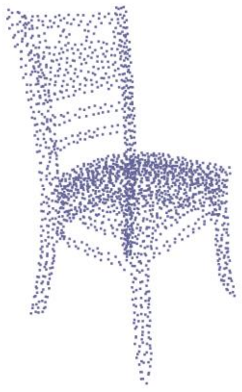
Figure 7: Original point cloud

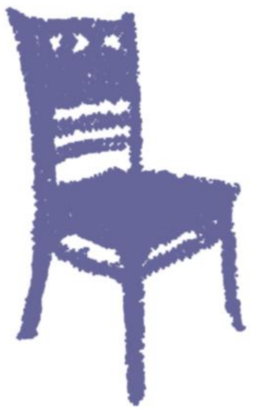
Figure 8: Upsampled point cloud

PCL also implements many basic statistical metrics that can be used to describe aspects of a point cloud. These include metrics like bounding box size, location of a point cloud's centroid, and clustering algorithms. One of these common metrics is known as nearest neighbor average distance. This metric samples one point, measures the distance to its nearest neighbor in the point cloud, repeats this process for every point in the point cloud, and averages all these distances. Nearest neighbor average distance is calculated using Equation 1, where $\overline{NND_P}$ represents the nearest neighbor average distance of a point cloud P and $|P|$ represents the number of points in the point cloud P.

$$\overline{NND}_{\mathrm{P}} = \frac{\sum_{\mathrm{i}=1}^{|\mathrm{P}|} \text{Euclidean distance from } \mathrm{P_i} \text{ to } \mathrm{P_i}'\text{s nearest neighbor}}{|\mathrm{P}|} \tag{1}$$

Another common statistical analysis tool implemented in PCL is principal component analysis (PCA). This algorithm, which is well established in data science, can be used on a point cloud to find the three principal components of a point cloud, along with the percentage of variance accounted for by each of those principal components found. PCA is used frequently in point cloud research to align point clouds along their principal component axes [43,44].

While the PCL provides many useful point cloud post-processing tools, several competitors have worked to address the need to help users choose the appropriate post-processing tool(s). These competitors include MeshLab [45], a graphical package for processing mesh files that doesn't require knowledge of C programming, libigl [46], a C++ library with tools more focused on discrete differential geometry manipulation, rather than just point cloud manipulation, and cilantro [15], a lean, efficient C++ library created for general point cloud processing that aims to develop tools that are easy to understand.

In 2018, a team published an open-source Python library, known as Open3D [47], to make point cloud post-processing tools easier to use. Open3D established a clear review process for proposed features for the library, and a robust version control protocol. Open3D exposes a set of C++ data structures and algorithms in Python with the purpose of creating a fast, light, and easy to implement collection of point cloud post-processing techniques. However, Open3D's documentation provides no guidance on how to go about finding suitable parameters for these post-processing techniques.

Statistical Outlier Removal Filter

Open3D includes a statistical outlier removal filter that requires two input arguments called "number of neighbors" and "standard ratio". The 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. Formally, this process can be defined as Equation 2. In this equation, P_n represents the raw, unfiltered cloud, P_i represents any single point in P_n and P_f represents the resulting filtered point cloud. In addition, *x* represents the value given for the "number of neighbors" argument, and *y* represents the value given for the "standard ratio" argument.

$$P_f = \{P_n \mid The\ average\ distance\ of\ P_i\ to\ its\ nearest\ x\ number\ of\ neighbors\ is\ within\ y\ standard\ deviations\ of\ the\ overall\ distribution\ for\ P_n\} \tag{2}$$

This filter has been shown to reduce the number of outliers in a point cloud, but only when used with the correct parameters [48]. Finding these correct parameters requires an iterative approach, in which a user must repeatedly guess filter parameters until satisfactory filtration results are achieved. Point cloud data does not always have a ground truth point cloud that would provide a visualization for what the properly filtered point cloud should look like. Without a ground truth point cloud reference, a visual inspection usually needs to be performed after every filtration iteration. The repetitive process of guessing parameter values and visual inspection takes expertise and significant time to reduce outliers consistently and effectively maintain the ground truth surface. The only automated methods that don't require user provided values for filtration parameters have used supervised learning methods, which require large, labeled datasets and can be difficult to troubleshoot when performance is not satisfactory.

Multi-View Partial (MVP) Point Cloud Database

Parallel to the development of these advances in post-processing, several publicly available point cloud databases were created [49,50]. One of these databases, the MVP Point Cloud Database, was created to provide a benchmark for the point cloud community to use for various applications, including point cloud completion, point cloud registration, and other standard processes [51]. This database consists of 2,400 point clouds of varying geometry, each with 2,048 uniformly sampled points across the point cloud. An example of a point cloud from this database is shown in Figure 9.

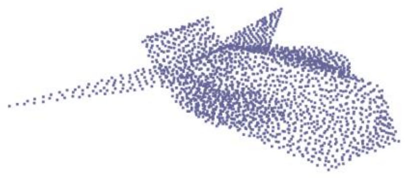

Figure 9: A point cloud of a ship from the MVP point cloud database

Synthetic Noise Generation

Although many point cloud research databases exist, there is a lack of standard databases of point clouds with noise and outliers. Research teams have investigated point cloud denoising by adding synthetic noise and outliers to point clouds [16,8]. These synthetic points clouds can be used to test filtering techniques. Research teams take slightly different approaches to creating synthetic point cloud databases. However, in general, the process consists of taking some ground truth point cloud and corrupting the ground truth point cloud by adding Gaussian noise to that point cloud. This Gaussian noise is scaled by some metric associated with the overall size of the point cloud.

Real World Point Cloud Research Databases

In addition to creating synthetic point clouds, several databases have been recently introduced that include real world point clouds with noise and outliers. One of these databases, introduced by Wolff et al., was collected using photogrammetry [19]. One of these point clouds is shown in Figure 10. In addition, Figure 11 shows a filtered version of this point cloud provided in the reference. Using this real-world data, teams have worked to ensure that investigated filters work not only on synthetic data, but also on real world data [16]. These real-world databases are needed because the distribution of noise and outliers that a 3D scanner generates can be difficult to simulate in artificially generated point clouds [52].

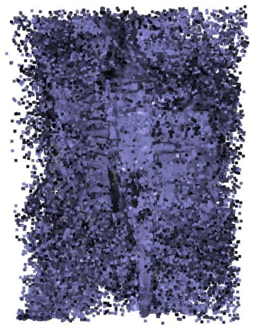

Figure 10: A noisy photogrammetry point cloud presented by Wolff et al.

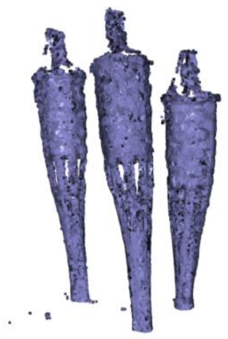

Figure 11: The filtered version of this photogrammetry point cloud

Point Cloud Comparison Metrics

Research groups have created several metrics to quantitatively measure the visual difference between two point clouds [53,54]. These metrics numerically analyze the performance of a filter, rather than using subjective visual approaches. Visual comparison metrics cannot be used to automatically find suitable filtration parameters in the real-world, because they require a ground truth point cloud to measure from.

The chamfer distance, a measure created in previous research, is an efficient way to determine the dissimilarity between two multidimensional pointsets [55]. Chamfer distance is a numerical measurement for the visual difference between two point clouds (e.g., point clouds “A” and “B”) [16,51]. For each point in point cloud A, a corresponding point in point cloud B is found, such that the Euclidean distance between the two points is a minimum. This distance is then squared and added together, for every point in point cloud A. This process is repeated for point cloud B, and finally, the two sums are added together. Chamfer distance is calculated using Equation 3 [16,56], where P_f represents the set of all points in the filtered point cloud, and P_o represents the set of all points in the original point cloud. There is no perfect metric for measuring the similarity between two point clouds, but chamfer distance provides a computationally efficient method that is widely used by the point cloud research community[56,57].

$$D_c = \Sigma_{p' \in P_f}(\text{distance from p' to its nearest neighbor in } p_o)^2 + \Sigma_{p \in P_o}\left(\text{distance from p to its nearest neighbor in } p_f\right)^2 \tag{3}$$

3D scanning is a technology that has been in a constant state of improvement for the last several decades. While growing from government funded research to critical technology in many industries, users have created significant infrastructure to enable the use of these files. This infrastructure includes new filetypes, advances in post-processing, and libraries that make these post-processing advances easier to implement. Additionally, research teams have created many standardized databases of both artificial and real-world point clouds to test new post-processing methods on.

These past research efforts have resulted in 3D scanning and point cloud technology that is far more accurate and reliable than it was just a decade ago. However, there are still areas of potential improvement that need to be addressed. From point cloud filters whose filtration performance does not rely on user input, to standardized databases for the entire community to work from, there are many topics that would benefit from further investigation.

RESEARCH OBJECTIVES

The primary goal of this research was to study the potential of automatically determining appropriate filtration parameter values for Open3D's statistical outlier removal filter. Specifically, the following research tasks were undertaken:

1. *Developed a point cloud comparison metric to supplement the chamfer distance:* To support the validity of using the chamfer distance as a comparison metric, a summary statistic comparison metric was created to supplement the chamfer distance, and their correlations were analyzed.
2. *Studied two different approaches that determine appropriate parameters for Open3D's statistical outlier removal filter without requiring user input:* Two different approaches will be presented that find arguments for the "number of neighbors" parameter and the "standard ratio"

parameter that are likely to result in a well filtered point cloud that does not require user iteration.

3. *Assessed the performance of these approaches quantitatively and qualitatively on synthetic and real-world point clouds:* The performance of these presented approaches were tested on synthetic point clouds taken from the Multi-view Partial Point Cloud Database and real-world point clouds from the dataset presented by Wolff et al.[19]

METHODOLOGY

This section outlines the approaches used to filter point clouds without requiring user input and the methods used to investigate the effectiveness of point cloud outlier removal without user input. Artificial and real-world point clouds were used to evaluate the rigor and reliability of the presented approaches. A sensitivity analysis was performed to determine how robust these approaches were against small perturbations in the parameters that were automatically determined. For each artificial point cloud that was filtered using these automatic approaches, a percentage error was found between the automatically filtered point cloud and an estimate of the best filtration results possible. The real-world filtered point clouds were evaluated from both a qualitative and a quantitative perspective to remove as much subjectivity as possible in the performance assessment. The approaches used to automatically find acceptable point cloud filtration parameters developed by the authors use the Open3D statistical outlier removal filter. A viable method of automatically determining values for the “number of neighbors” and “standard ratio” parameters is defined as one that removes the need for any iteration or user input, while producing visually and numerically cleaner point clouds.

Automatic Filter Parameter Estimation

The two approaches used to determine acceptable parameters for Open3D’s statistical outlier removal filter plotted the statistical outlier removal filter’s parameters. These parameters were plotted against two different metrics of the resulting filtered point cloud. The first of these metrics was the percentage of the filtered point cloud’s total variance accounted for by each principal component. In PCA, each principal component models the original variables in the dataset information into single indices. The output is an uncorrelated set of unique principal components that avoids including the same information. Each index provides unique information from the original variables even when they were highly correlated. The principal components were found by using scikit-learn, a common ML and data analysis library for Python [59]. After finding these components, the one that accounted for the smallest percentage of variance in the cloud was isolated. This percentage, called the "principal component least variance ratio" was then plotted against the two parameters of the statistical outlier removal function, using matplotlib's 3D plot feature [60]. An example of the surface created, showing the relationship between a filtered point cloud's principal component least variance ratio and the two statistical outlier removal filter parameters, can be seen in Figure 12.

For the surface created, the minimum was found, and the "number of neighbors" and "standard ratio" associated with that minimum were used as parameters for Open3D's statistical outlier removal filter. This filter was used to produce a final, filtered point cloud by using these filtration parameters at this minimum location to clean the original, unfiltered point cloud. In Figure 12, a red star indicates the minimum of the surface plotted.

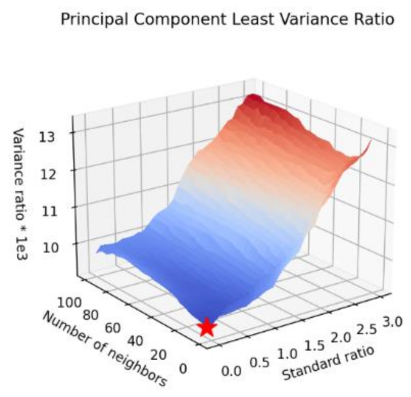

Figure 12: Surface created in the statistical outlier removal parameter graphing, with a red star indicating the plotted surface's minimum

The second metric investigated was the nearest neighbor average distance. A function was created to plot the statistical outlier removal filter parameters against this average distance. This surface was then plotted using the same tools and techniques to plot the principal component least variance ratio surface described above. Again, the parameters associated with the minimum of this surface were used to filter the original, unfiltered point cloud.

It was hypothesized that by automatically selecting filtration parameters that are associated with the lowest principal component least variance ratio or the lowest nearest neighbor average distance, an optimal, or near-optimal, filtered point cloud would be generated. For the principal component least variance ratio, this was believed for a few reasons. First, principal component analysis is a statistical measure that does not depend on a point cloud's orientation. So, variance measures computed would be independent of a point cloud's orientation. Second, principal component analysis is a function of a point cloud's geometry, meaning that components computed would represent variances based on geometric features of the point cloud. Lastly, selecting the principal component with the smallest variance indicates the point cloud with the tightest collection of points, or one with the fewest outliers. Plotting the least variance ratio against the two parameters of the statistical outlier removal function creates a 3D design space. If optimized to find the minimum solution in this space, the resulting parameters should produce a point cloud with the most outliers removed.

Similarly, it was thought that selecting parameters associated with the lowest nearest neighbor average distance would result in removing points that are outliers, while maintaining ground truth points. This was thought because the removal of these ground truth points would result in a subsequent increase in nearest neighbor average distance.

Synthetic Noise Generation

To evaluate the automated point cloud filtration approaches proposed using the MVP point cloud database, a point cloud taken from the MVP Point Cloud Database was corrupted using an algorithm similar to past research [20]. The algorithm used to generate synthetic noise points and outliers in this research work used a magnitude multiplier parameter and a corruption percent parameter. This corruption algorithm randomly selected a percentage of points in the input point cloud, equal to the corruption percent parameter, and moved these points in a random direction. The extent of this movement was determined using a normal distribution, whose average was zero and whose standard deviation, σ, was calculated using Equation 4.

$$\sigma = \tau * \sqrt{dx^2 + dy^2 + dz^2} \qquad (4)$$

In Equation 4, *dx, dy,* and *dz* are the sizes of the original point cloud's bounding box in the x, y, and z directions, respectively, and τ is the magnitude multiplier parameter.

Multiple values for the magnitude multiplier and corruption percentage parameters were used to evaluate if outliers in a point could be effectively removed without requiring user input. These values were used to simulate different magnitudes of outlier points, both in distance from their ground truth surface, and their quantity. Values were chosen in such a way as to evaluate this automated approach's performance over a wide span of both outlier intensity and frequency. The parameters used for this corruption are shown in the results section. An image of a point cloud with noise and outliers generated through this process, (referred to as a "corrupted cloud" for the rest of this paper), can be seen in Figure 13. For this figure, a magnitude multiplier and corruption percentage argument of 0.05, and 0.2 respectively were used to generate outliers.

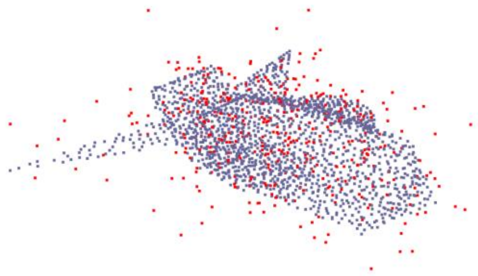

Figure 13: A corrupted point cloud generated from the MVP database using a magnitude multiplier and corruption percentage argument of 0.05 and 0.2, respectively, with generated noise and outliers shown in red

Automatic Filter Performance Measurement

To quantitatively measure the performance of these automated approaches, a new metric was created to assess how well these approaches filtered a synthetically corrupted MVP point cloud. This metric used was called the summary statistic. The summary statistic was defined using Equations 5-9. The definition of RMSE, recall, precision, and accuracy in the following equations are derived from past research [60], [17]. The summary statistic was defined so that it reduced all four performance indicators down to one, easily interpretable number. In the RMSE equation (5), P_f and $|P_f|$ represent the resulting filtered MVP point cloud, after being corrupted, and the number of points in the point cloud, respectively. Additionally, P_o represents the ground truth MVP point cloud. If a point $p_i' \notin P_f$, but $p_i \in p_o$, because of p_i being filtered out, that index *i* is not used in the RMSE calculation.

$$\text{RMSE} = \frac{\sqrt{\frac{\Sigma_{p_i' \in P_f p_i \in P_o} (\text{distance between } p_i \text{ and } p_i')^2}{|P_f|}}}{\text{nearest neighbor average distance in original cloud}} \quad (5)$$

$$Precision = \frac{|\text{noise and outlier points removed by filter}|}{|\text{points removed by filter}|} * 100 \quad (6)$$

$$Recall = \frac{|\text{noise and outlier points removed by filter}|}{|\text{noise and oulier points in original noisy point cloud}|} * 100 \tag{7}$$

$$Accuracy = \frac{|\text{noise and outlier points removed by filter}| + |\text{correct points that are left}|}{|\text{points in original point cloud}|} * 100 \tag{8}$$

$$Summary\ statistic\ = \frac{avg(\text{precision, recall, accuracy})}{\text{RMSE}} \tag{9}$$

Precision, recall, and accuracy, numbers that have better performance associated with higher values, make up the numerator of this summary statistic, while the RMSE forms the denominator. This definition results in a metric whose value is increased by higher accuracy, precision, and recall values, along with lower RMSE values. In addition, by averaging the accuracy, precision, and recall, this ensures that no singular metric has more weight than any other metric. Ultimately, this metric works well to evaluate the effectiveness of a point cloud filtration process because it collects information regarding a point cloud filter’s ability to remove outliers, while maintaining ground truth points, and represents this metric using one number, where higher values indicate better filtration performance.

The summary statistic was used to assess filter performance by finding the percentage error between the summary statistic of the automatically filtered MVP point cloud and the summary statistic of the filter using the best possible performing parameters for the statistical outlier removal filter. These best possible parameters were found by selecting the filtration parameters that resulted in a maximum for the summary statistic. This maximum summary statistic was found by creating a 32 x 32 surface, where each point on the surface was associated with some combination of filtration parameters. At each point of this surface, the summary statistic of the filtered point cloud that resulted from filtering the corrupted point cloud with those associated filtration parameters was measured and then plotted.

This 32 x 32 resolution was chosen because increasing this resolution by a factor of two along both axes changed the summary statistic error by only 1.9%, for the minimum nearest neighbor average distance approach, averaged over ten point cloud filtrations. Similarly, for the minimum principal component least variance ratio approach, percentage errors were only changed by .2%, but overall calculation time increased by 387%. Due to these reasons, a 32 x 32 resolution was deemed to be sufficient.

Each point on this surface was associated with some filtered MVP point cloud. An example image of the surface created in this process is shown in Figure 14, with the maximum summary statistic indicated by a blue star. Although this maximum is at the edge of the plotted surface, it can be known that this is at least a local maximum due to the fact that the summary statistic is not defined for a non-positive number of neighbors parameter, because the statistical outlier removal filter doesn't work with a non-positive number of neighbors.

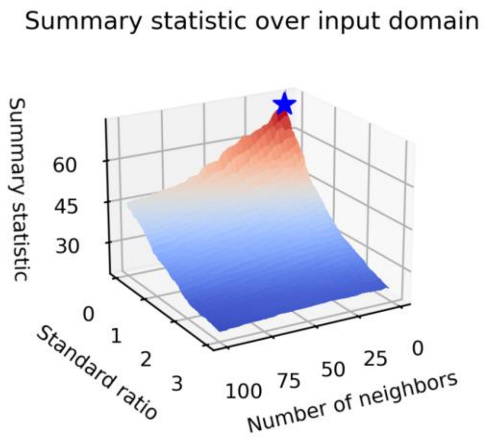

Figure 14: An example of the search process carried out for finding the maximum possible summary statistic, with the maximum summary statistic indicated by a blue star

Similarly, the chamfer distance was also used to measure the performance of this automated approach, evaluating the difference between a filtered MVP point cloud and a ground truth MVP point cloud before it was corrupted. As a result of using synthetic point clouds, alignment and registration was not a concern when measuring chamfer distance. The chamfer distance was used to assess filter performance in an almost identical manner to the summary statistic, however, the best possible parameters were found using the minimum of the chamfer distance, rather than the maximum.

Evaluation of Automatic Parameter Selection

The proposed approaches were each evaluated for performance. First, a point cloud from the MVP database was randomly selected. This point cloud was then corrupted using the process described in the Synthetic Noise Generation section. Finally, this point cloud was then processed with the statistical outlier removal parameter graphing functions described in the Automatic Filter Parameter Estimation. The generated surfaces were created using a number of neighbors argument with a range of [1, 50], with ten points generated in this range, in a linear distribution. Additionally, the standard deviation axes were created using a range of [0.000001, 3] with ten points in this range, in a linear distribution.

These ranges of values used to generate the surfaces were selected because they were the only values that consistently had a local minimum. New local minimum locations were not produced on these surfaces, when testing larger ranges, but calculation times increased. Increasing the domain for the number of neighbors and standard ratio from [1, 100] and [0.000001, 3] to [1,200] and [0.000001, 6] respectively, with the same surface resolution, increased calculation time by 20%, averaged over ten point cloud filtrations. In Figure 15, the surface’s minimum, marked with a red star, is the same as the minimum indicated in Figure 16. The surface shown in Figure 16 provides no additional useful information but requires significantly more time to calculate a local minimum.

After generating these surfaces, the corrupted point cloud was filtered twice. This was done using two different sets of parameters acquired from the minimum nearest neighbor average distance and the minimum principal component least variance ratio surfaces described in the Automatic Filter Parameter

Estimation section. These two filtered point clouds were then compared to an optimally filtered point cloud. This comparison method followed the summary statistic and chamfer distance metrics described in the Automatic Filter Performance Measurement section.

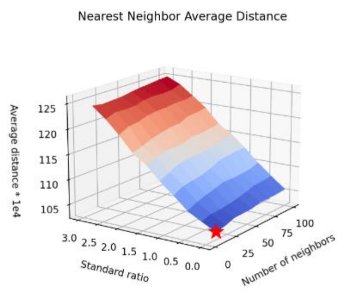

Figure 15: Nearest neighbor average distance surface plotted over a properly sized domain, with the minimum indicated by a red star

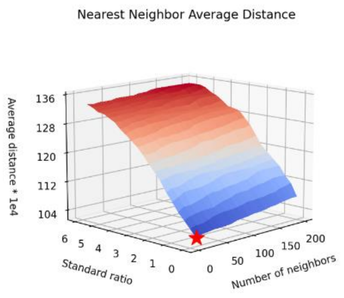

Figure 16: Nearest neighbor average distance surface plotted over a domain that shows no additional useful information

The process of randomly selecting an MVP point cloud, corrupting it, automatically filtering it, and comparing the resulting filtered point cloud to an optimally filtered cloud, was repeated ten times for a randomly selected MVP point cloud. The acquired chamfer distance and summary statistic percentage errors were averaged, and the process was repeated for 49 additional randomly selected MVP point clouds. These percentage errors were again averaged together. In total, 50 randomly selected MVP point clouds were corrupted and then automatically filtered and evaluated, using both the minimum principal component least variance ratio approach and the minimum nearest neighbor average distance approach. Analysis of these 50 MVP point clouds was repeated five separate times with different amounts and magnitudes of synthetically added noise and outliers, to investigate how these automatic approaches performed under various conditions.

Three uncorrupted MVP point clouds were selected from the set of 50 point clouds used in the above experimentation. These point clouds were selected so that when using the minimum nearest neighbor average distance approach to automatically find filtration parameters for their corrupted counterparts, the recorded chamfer distance percent errors and summary statistic percent errors were at least one standard deviation below the average for the 50 MVP point clouds tested. The process of corrupting an MVP point cloud, automatically filtering, and measuring performance was repeated with these three selected MVP points clouds using six different resolutions for the average nearest neighbor distance surface created.

The six different resolutions used were 5 x 5, 7 x 7, 10 x 10, 14 x 14, 22 x 22, and 32 x 32 surfaces. These resolutions were chosen to investigate the sensitivity of the parameters found at the surface's minimum, and to determine how precise the surface needs to be to achieve acceptable results. Figures 17 and 18 show two different example surfaces created using a 5 x 5 surface (Figure 17), and a 32 x 32 surface (Figure 18). On both these surfaces, a red star indicates the minimum of the plotted surface. This process of corrupting and filtering three MVP point clouds that filtered well using the minimum nearest neighbor average distance approach was then repeated, using point clouds that the minimum principal component least variance ratio approach performed well on.

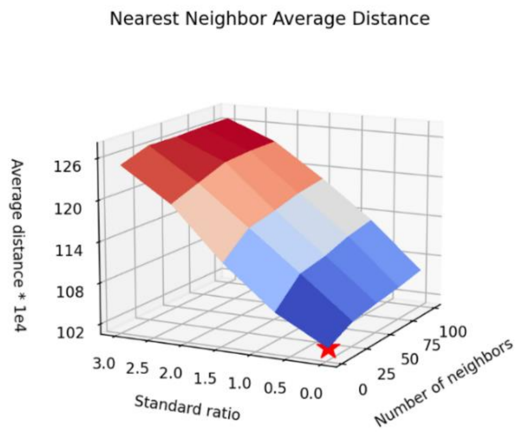

Figure 17: An example of a 5 x 5 surface for the nearest neighbor average distance, with the surface minimum indicated with a red star

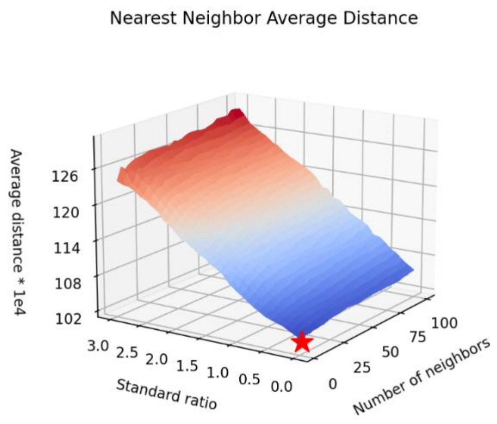

Figure 18: An example of a 32 x 32 surface for the nearest neighbor average distance, with the surface minimum indicated with a red star

Following these sensitivity analysis tests, two of the uncorrupted MVP point clouds used for these tests were selected. One was selected from the minimum nearest neighbor average distance MVP point clouds, and one was selected from the minimum principal component least variance ratio MVP point clouds. These clouds were upsampled using the moving least squares algorithm from PCL. An identical sensitivity analysis was carried out for these two point clouds, and the results were compared to their standard, 2,048-point variants. This process allowed for the evaluation of these automated approaches performance on dense point clouds.

Finally, in order to evaluate the performance of these automated approaches on real world data, two photogrammetry point clouds presented by Wolff et al. [19] were filtered using both the minimum principal component least variance ratio and the minimum nearest neighbor average distance approach. These filtered point clouds were compared to filtered point clouds presented by Wolff et al., on both a qualitative and quantitative basis. This process allowed for the analysis of whether these automated approaches continue to work on real-world outliers.

RESULTS

Tables 1 and 2 show the average results from the first experimental process of varying the amount of noise and outliers added to 50 randomly selected MVP database point clouds, and then filtering them using these proposed automated approaches. Each row in either table represents the results of 500 experimental runs, using 50 different point clouds. In total, each table shows the averaged results from 2,500 experimental runs, using either the minimum nearest neighbor average distance or the minimum principal component least variance ratio approach to filter the corrupted points cloud, resulting in 5,000 total experimental runs. Upon inspecting these tables, these automatic approaches have extremely high error percentages when used on point clouds with a small amount and magnitude of outliers. However, as the magnitude and number of outliers increase in a corrupted MVP point cloud, the average percentage error drops precipitously. Finally, the error percentage in the summary statistic generally follows the direction of the error in the chamfer distance, however the amount of change between different levels of noise is much greater for the chamfer distance error percentage.

Table 1: Filtration results from 50 randomly sampled MVP point clouds, using the minimum nearest neighbor average distance approach

Corruption Percentage	Magnitude Multiplier	Average Summary Statistic Error	Summary Statistic Standard Deviation	Average Chamfer Distance Error	Chamfer Distance Standard Deviation
30%	.05	4.17%	2.06%	30.86%	25.32%
20%	.1	6.12%	3.00%	22.22%	40.64%
20%	.05	8.17%	3.65%	70.96%	42.79%
20%	.01	7.66%	3.76%	214.17%	115.74%
10%	.05	15.07%	5.21%	249.37%	154.88%

Table 2: Filtration results from 50 randomly sampled MVP point clouds, using the minimum principal component least variance ratio approach

Corruption Percentage	Magnitude Multiplier	Average Summary Statistic Error	Summary Statistic Standard Deviation	Average Chamfer Distance Error	Chamfer Distance Standard Deviation
30%	.05	18.92%	11.22%	130.77%	111.09%
20%	.1	20.94%	10.90%	166.12%	316.79%
20%	.05	25.25%	12.22%	226.09%	257.62%
20%	.01	14.96%	10.26%	322.67%	271.33%
10%	.05	25.98%	9.02%	420.04%	270.08%

Figures 19 – 22 show images of several point clouds throughout the stages of this process. The corrupted point clouds have a corruption percentage of 20%, and a multiplier magnitude of 0.05. Evident in this figure is the reduction of outliers by all approaches used. Specifically, the transition from Figure 19 to Figure 20 shows only surface level noise left in the filtered point cloud.

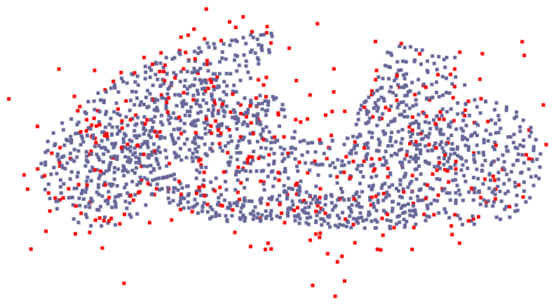
Figure 19: Corrupted MVP point cloud of a motorcycle to be filtered using the minimum nearest neighbor average distance approach

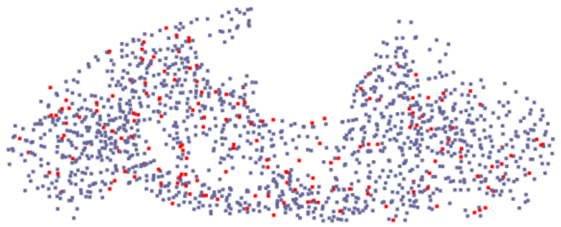
Figure 20: Filtered MVP point cloud of a motorcycle, filtered using the minimum nearest neighbor average distance approach

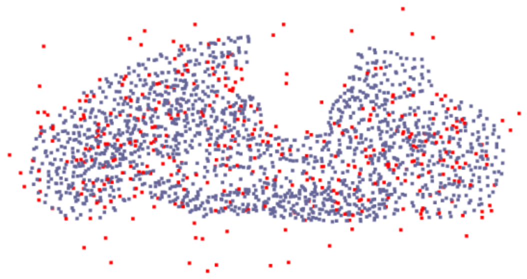
Figure 21: Corrupted MVP point cloud of a motorcycle to be filtered using the minimum principal component least variance ratio approach

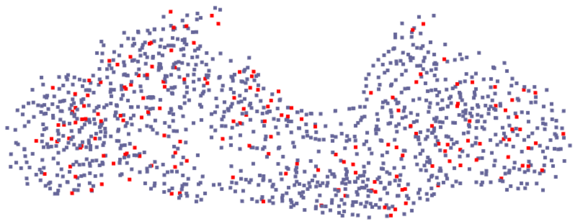
Figure 22: Filtered MVP point cloud of a motorcycle, filtered using the minimum principal component least variance ratio approach

A sensitivity analysis was then performed for the minimum nearest neighbor average distance approach. Three MVP point clouds, shown in Figures 23 - 25, were selected so that the minimum nearest neighbor average distance approach filtered by one standard deviation better than the average, across the 50 MVP point clouds, as described in the Evaluation of Automatic Parameter Selection section.

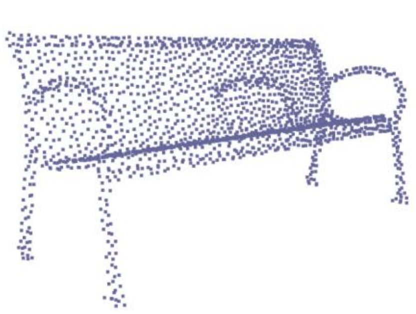

Figure 23: Point Cloud 1

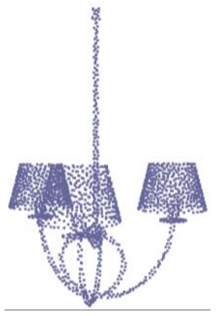

Figure 24: Point Cloud 2

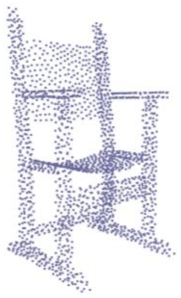

Figure 25: Point Cloud 3

The results of the sensitivity analysis for the point clouds shown in Figures 23 – 25 are shown in Figures 26 – 28. This sensitivity analysis used noise generation settings of 20% and 0.05 for the corruption percentage and magnitude multiplier arguments respectively. Figures 26 - 28 show that the chamfer distance is affected by surface resolution more than the summary statistic. At resolutions below 14 x 14, the chamfer distance percentage error is above 40%. The summary statistic percentage error remained nearly below 20% at the lowest (5 x 5) surface resolution. These results indicate that when using the minimum nearest neighbor average distance approach, a surface resolution of at least 14 x 14 is needed to achieve consistent results.

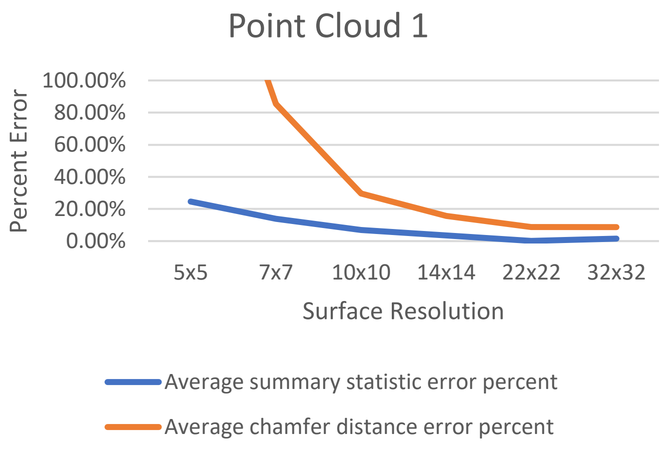

Figure 26: Sensitivity analysis results for Point Cloud 1

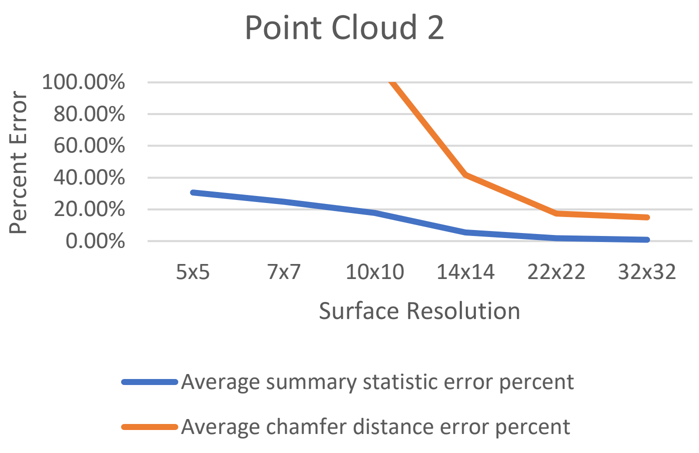

Figure 27: Sensitivity analysis results for Point Cloud 2

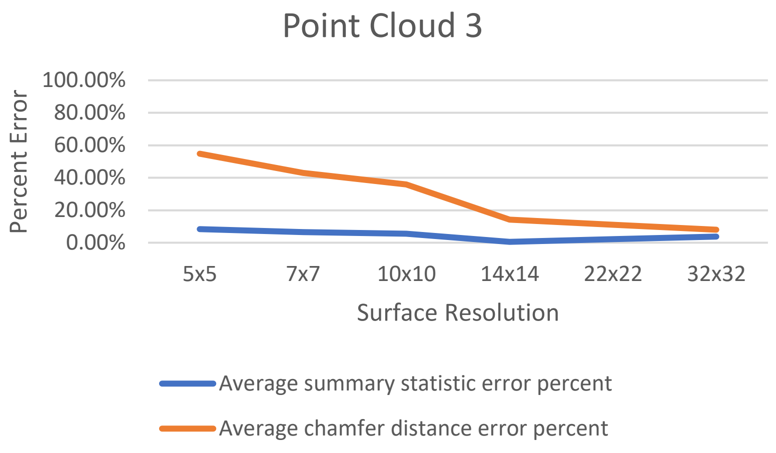

Figure 28: Sensitivity analysis results for Point Cloud 3

Figures 29 - 32 show two examples of this sensitivity analysis being carried out on Point Cloud 3. Each row is associated with a specific nearest neighbor average distance surface resolution. These figures reveal several insights concerning this sensitivity analysis. The first of these insights, seen in Figure 30, is that at a low surface resolution, the outlier removal parameters found using the minimum nearest neighbor average distance approach seem to be too aggressive, creating a large hole in the back of the chair. Additionally, seen in Figure 32, the filtered point cloud has smaller holes present in its filtered surface.

This aligns with the findings shown in Figure 28, which shows a more accurately automatically filtered point cloud at a resolution of 10 x 10.

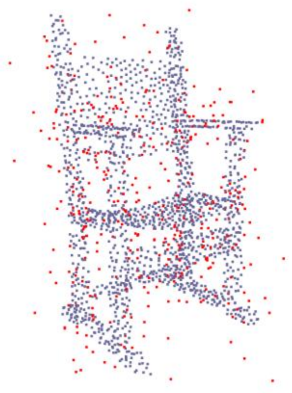
Figure 29: A corrupted Point Cloud 3, to be filtered using the minimum nearest neighbor average distance approach, at a surface resolution of 5 x 5

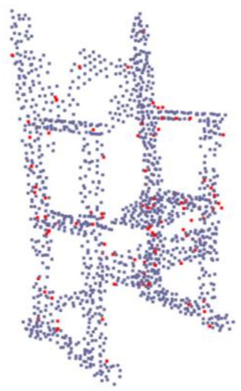
Figure 30: A filtered Point Cloud 3, filtered using the minimum nearest neighbor average distance approach, at a surface resolution of 5 x 5

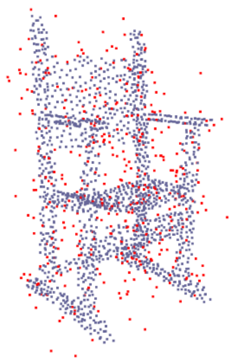
Figure 31: A corrupted Point Cloud 3, to be filtered using the minimum nearest neighbor average distance approach, at a surface resolution of 10 x 10

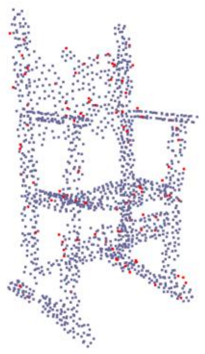
Figure 32: A filtered Point Cloud 3, filtered using the minimum nearest neighbor average distance approach, at a surface resolution of 10 x 10

Following this sensitivity analysis of the minimum nearest neighbor average distance approach, this same analysis was carried out for the minimum principal component least variance ratio approach. Three MVP point clouds, shown in Figure 33 – 35 were chosen such that, when using the minimum principal component least variance ratio approach, these point clouds were filtered by one standard deviation better than the average, across the 50 MVP point clouds, as described in the Evaluation of Automatic Parameter Selection section.

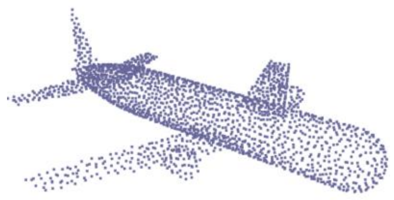
Figure 33: Point Cloud 4

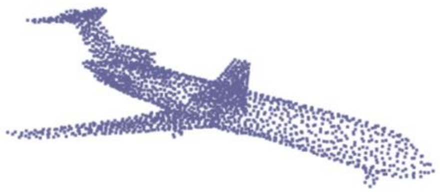
Figure 34: Point Cloud 5

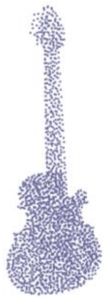
Figure 35: Point Cloud 6

These three MVP point clouds were the subject of the sensitivity analysis for the minimum principal component least variance ratio approach, whose results are shown in Figures 36 - 38. This sensitivity analysis used the same noise generation settings as for the minimum nearest neighbor average distance sensitivity analysis. As can be seen in Figures 36 - 38, for all three point clouds analyzed, there was a very weak correlation between the percent error in the chamfer distance and the surface resolutions used for the minimum principal component least variance ratio approach. The summary statistic follows a similar pattern. This suggests that, when using the minimum principal component least variance ratio approach (Figures 36 – 38) to filter point clouds, the parameters found are much less susceptible to slight perturbations in their values, compared to the minimum nearest neighbor average distance approach (Figures 26 – 28).

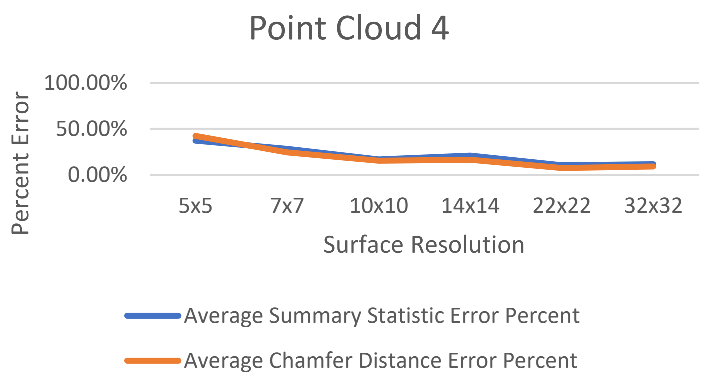

Figure 36: Sensitivity analysis results for Point Cloud 4

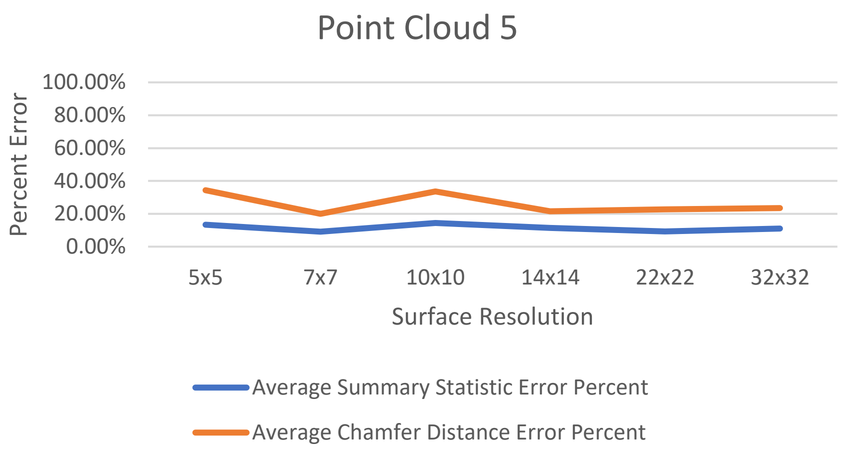

Figure 37: Sensitivity analysis results for Point Cloud 5

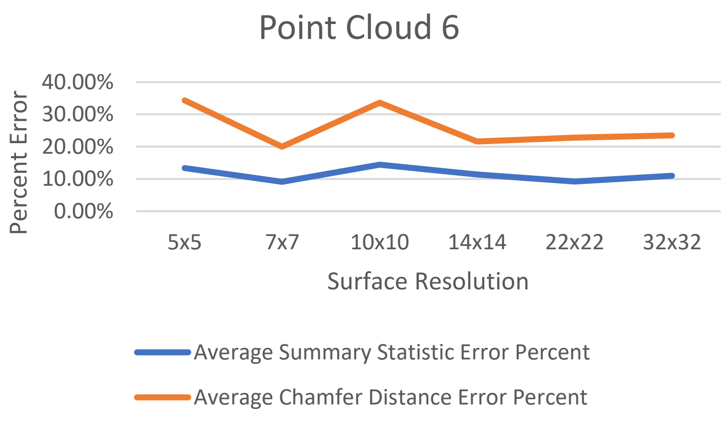

Figure 38: Sensitivity analysis results for Point Cloud 6

In addition to these sensitivity analyses above, it was necessary to investigate these automated filtration approaches' performance on corrupted point clouds that have significantly more than 2,048 points. The sensitivity analysis method used for investigating both the minimum nearest neighbor average distance approach and the minimum principal component least variance ratio approach was carried out for two upsampled point clouds that were used in the above sensitivity analyses. These two point clouds were first upsampled using the PCL's moving least squares upsampling method. The upsampled point clouds, along with their number of points, are shown below, in Figure 39 and 40.

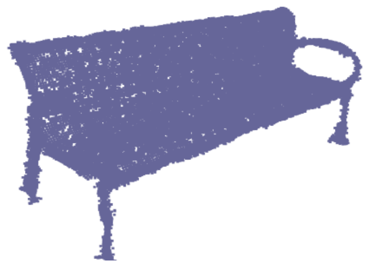
Figure 39: Point Cloud 1 upsampled with 73699 points

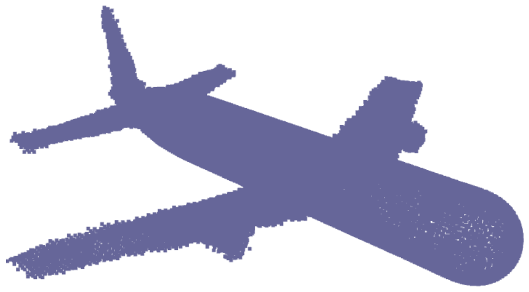
Figure 40: Point Cloud 4 upsampled with 72744 points

The results of the sensitivity analysis carried out on these two point clouds are shown in Figure 41 and 42. For this analysis, an upsampled MVP point cloud was corrupted with 20% and 0.05 for the corruption percentage and magnitude multiplier arguments respectively and was then filtered using one of the automated approaches developed in this work. Specifically, the minimum nearest neighbor average distance approach was used for the upsampled version of Point Cloud 1, and the minimum principal component least variance ratio approach was used for the upsampled version of Point Cloud 4. Figure 41 shows that when using the minimum nearest neighbor average distance approach, there is not much of a correlation between percentage error and surface resolution, until the 22 x 22 resolution, at which point the chamfer distance error drastically increases. One possible cause for this is that this approach is potentially "overfitting" when such high resolutions are used, in combination with dense point clouds.

The performance of the minimum principal component least variance ratio approach continues to have very little correlation with the surface resolution.

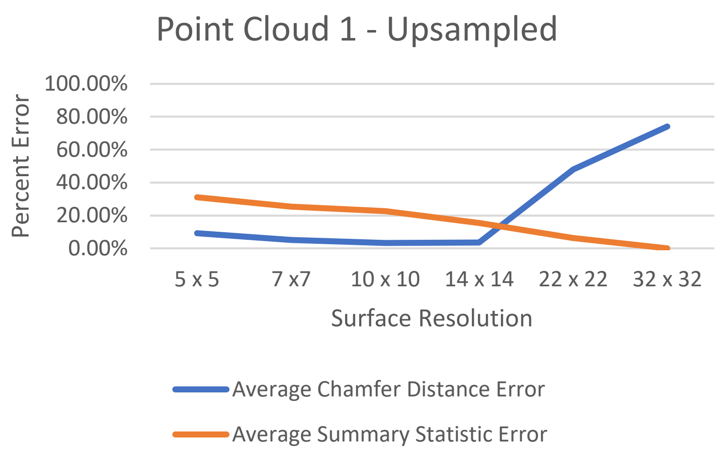

Figure 41: Sensitivity analysis results for an upsampled Point Cloud 1

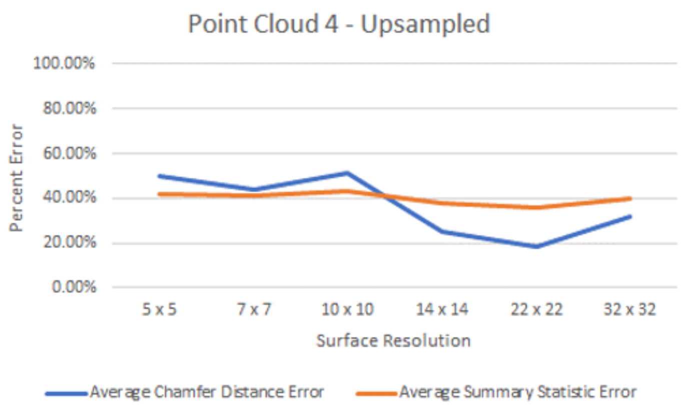

Figure 42: Sensitivity analysis results for an upsampled Point Cloud 4

Figures 43 - 46 show two examples of this process being carried out on the upsampled version of Point Cloud 1. Each row is associated with a specific nearest neighbor average distance surface resolution. This figure reveals several insights concerning this sensitivity analysis. One of these insights, seen by comparing Figure 44 and 46 is that the surface resolution appears to have very little effect on the effectiveness of the automated point cloud filtration approach. Between these two filtered point clouds, there's not a clear difference in quality. Another insight is the fact that this conclusion appears to be contradicted by the graph shown in Figure 41, indicating that at a surface resolution of 32 x 32, the automatically filtered point cloud should be of noticeable lower quality than that of the 10 x 10 surface filtered point cloud. This implies that chamfer distance may not be an appropriate metric to assess visual quality when highly dense point clouds are used. Additionally, by comparing Figures 44 and 46 to Figures 30 and 32, significantly more surface level noise is missed by these automatic outlier filtration approaches, when used on highly dense point clouds.

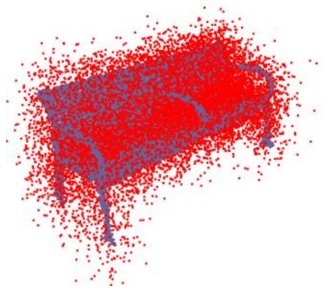

Figure 43: Corrupted upsampled version of Point Cloud 1, to be filtered using the minimum nearest neighbor average distance approach, with a surface resolution of 5 x 5

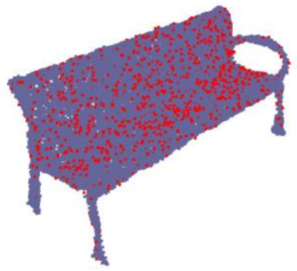

Figure 44: Filtered upsampled version of Point Cloud 1, filtered using the minimum nearest neighbor average distance approach, with a surface resolution of 5 x 5

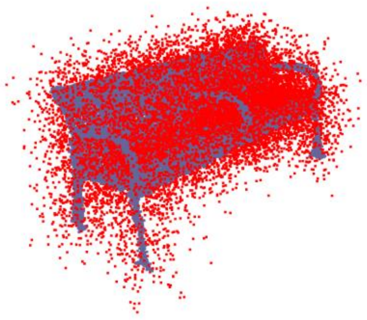

Figure 45: Corrupted upsampled version of Point Cloud 1, to be filtered using the minimum nearest neighbor average distance approach, with a surface resolution of 32 x 32

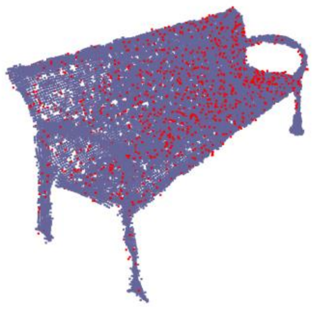

Figure 46: Filtered upsampled version of Point Cloud 1, filtered using the minimum nearest neighbor average distance approach, with a surface resolution of 32 x 32

Finally, these automatic outlier filtration approaches were tested on two real world point clouds presented by Wolff et al. [19]. This was done to assess whether the performance of these automatic approaches to finding filtration parameters maintain their performance when used on real-world outliers. Shown below, in Figures 47 and 48, are the unfiltered real-world point clouds presented by Wolff et al. The two photogrammetry point clouds filtered are referred to as “torches” and “statue.”

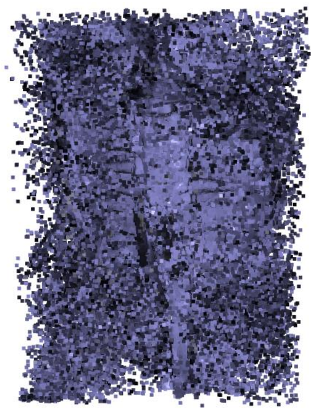
Figure 47: Original photogrammetry torches point cloud

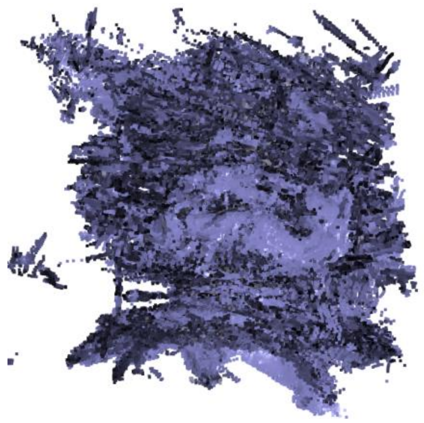
Figure 48: Original photogrammetry statue point cloud

These real-world point clouds were filtered using both the minimum nearest neighbor average distance approach and the minimum principal component least variance ratio approach. These filtered point clouds were compared to filtered point clouds presented by Wolff et al. In Figures 49 - 54 are images of these filtered point clouds, compared to point clouds that were filtered using these automatic parameter selection techniques. Visually, it appears that the minimum nearest neighbor average distance approach did at least as well as Wolff’s filter at filtering these point clouds. This can be seen in Figure 49, compared to Figure 51, where Figure 49, filtered using the minimum nearest neighbor average distance approach, has fewer holes in the surface left by the filtration when compared to Wolff’s filter in Figure 51. The minimum principal component least variance ratio approach clearly did not filter the point clouds as well as the other two filtration methods.

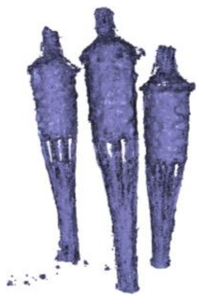
Figure 49: Torches point cloud filtered with minimum nearest neighbor average distance approach

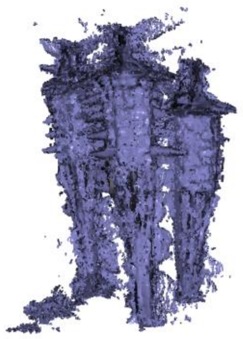
Figure 50: Torches point cloud filtered with minimum principal component least variance ratio approach

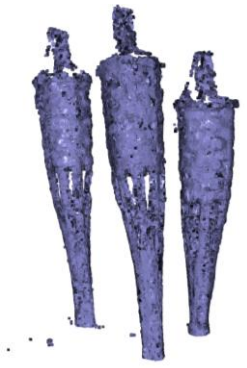
Figure 51: Torches point cloud filtered with Wolff's filter

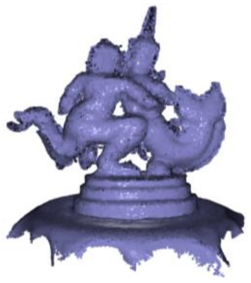
Figure 52: Statue point cloud filtered with minimum nearest neighbor average distance approach

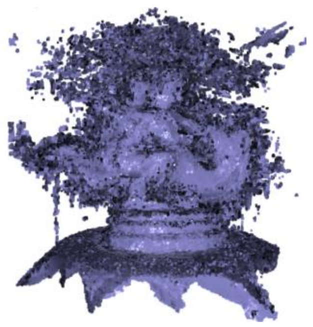
Figure 53: Statue point cloud filtered with minimum principal component least variance ratio approach

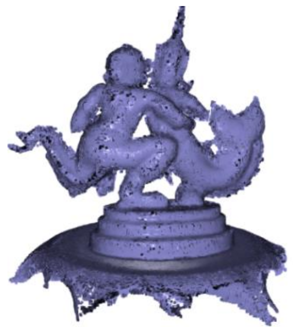
Figure 54: Statue point cloud filtered with Wolff's filter

The percentage of points removed from the original corrupted point clouds using these automatic approaches was compared to the percentage removed by Wolff's filter to allow for numerical analysis of these real-world results without the use of some ground truth point cloud. This information is shown in Table 3. When reviewing this table, it's obvious that Wolff's filter removed far more points than either of the automatic approaches to finding suitable filtration parameters. However, this does not necessarily mean that Wolff's filter performed better. From visual inspection of Figure 51 and 54, it appears that Wolff's filter may have been overly aggressive, leaving holes in the surface of both the torches point cloud and the statue point cloud

Table 3: Information regarding the percentage of points removed by each filtration technique

Point Cloud	Original Number of Points	Percentage of Points Removed by Minimum Nearest Neighbor Average Distance Approach	Percentage of Points Removed by Minimum Principal Component Least Variance Ratio Approach	Percentage of Points Removed by Wolff's Filter
Torches	3,604,726	28.08%	1.27%	69.56%
Statue	5,849,643	25.53%	2.52%	88.93%

Another reason for this large discrepancy in the number of points removed by Wolff's filter and by these automatic outlier removal approaches is surface level noise. This surface level noise highlights one major consideration of these proposed approaches to automated outlier removal, which is their inability to filter surface level noise. This means that the statue point cloud and the torches point cloud, whose corrupted surfaces have significant surface level noise, have many more points filtered out by Wolff's noise and outlier filter than by this exclusively outlier focused filtering approach.

Finally, it must also be considered that Wolff's filter utilizes both depth and RGB information as well, meaning that it has access to more information than these automated approaches do. This means that Wolff's filter can more accurately remove points that do not belong to the ground truth point cloud. However, a user does not always have information on a point's depth or RGB value. Depending on how a 3D scan was collected, a user may have only the point locations to work with. In these situations, if an automated approach to outlier removal is needed, the proposed minimum nearest neighbor average distance approach may suit their needs well, depending on the nature of the corrupted point cloud.

CONCLUSIONS

This paper investigated the validity of automatically determining appropriate parameters for Open3D's statistical outlier removal filter, to remove the need for user iteration. The proposed approaches use relationships between a point cloud's nearest neighbor average distance and principal component least variance ratio to find these parameters. Specifically, parameters are used such that both the nearest neighbor average distance and principal component least variance ratio were minimized. The effectiveness of these two approaches was investigated using the MVP point cloud database, synthetic noise generation, and two real-world photogrammetry point clouds presented by Wolff et al.

Through this investigation, it was concluded that both approaches tend to work better as the quantity and magnitude of outliers present in a corrupted point cloud increase. Additionally, the performance of the minimum nearest neighbor average distance technique is directly correlated with the resolution of the nearest neighbor average distance surface created. This contrasts with the minimum principal component least variance ratio approach not having a strong correlation with the principal component least variance ratio surface resolution. These findings were supported by a wide array of tests performed on these filtration techniques and continue to hold when using these techniques on real-world, photogrammetry point clouds.

It was found that the performance of the minimum neighbor average distance approach is comparable to past filters presented in research for point clouds that are high in outliers, but lags in performance when point clouds are high in surface level noise. It was also found that the performance of the minimum principal component least variance ratio approach does not perform as well or as consistently as the minimum nearest neighbor average distance approach to finding appropriate parameters for the statistical outlier removal filter.

This research also presents potential for future work, in further optimizing these automatic parameter determination approaches. Plotting a full grid of points every time a point cloud needs to be filtered is a very computationally intensive and inefficient task. Rather than naively plotting evenly spaced points across some predetermined domain, both these approaches could be redesigned to use gradient descent approaches. These approaches could be used to find the filtration parameters associated with both the minimum principal component least variance ratio and the minimum nearest neighbor average distance.

Ultimately, the approaches presented in this paper have potential to further streamline the post-processing of point clouds with heavy outlier occurrence. The minimum nearest neighbor average distance approach appears to hold more promise than the minimum principal component least variance ratio approach, but this does not mean that the latter approach is useless. If further investigation is done into the limitations and advantages of both these approaches, there is potential for the application of either of these approaches at a large scale, to quickly filter large amounts of point clouds without the need for user input or inspection.

BIBLIOGRAPHY

1. 3D Scanning Market to Rise at 10.2% CAGR till 2026. *GlobeNewswire* (2021).
2. Luo, S. & Hu, W. Score-Based Point Cloud Denoising. (2021).
3. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C. & Taubin, G. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. Vis. Comput. Graph.* **5**, 349–359 (1999).
4. Haleem, A. & Javaid, M. 3D scanning applications in medical field: A literature-based review. *Clin. Epidemiol. Glob. Health* **7**, 199–210 (2019).
5. Helle, R. H. & Lemu, H. G. A case study on use of 3D scanning for reverse engineering and quality control. *Mater. Today Proc.* **45**, 5255–5262 (2021).
6. Wang, Y. & Feng, H. Y. Effects of scanning orientation on outlier formation in 3D laser scanning of reflective surfaces. *Opt. Lasers Eng.* **81**, 35–45 (2016).
7. Charron, N., Phillips, S. & Waslander, S. L. De-noising of Lidar Point Clouds Corrupted by Snowfall. in *2018 15th Conference on Computer and Robot Vision (CRV)* 254–261 (2018). doi:10.1109/CRV.2018.00043.
8. Ning, X., Li, F., Tian, G. & Wang, Y. An efficient outlier removal method for scattered point cloud data. *PLOS One* (2018) doi:10.1371/journal.pone.0201280.
9. Stucker, C., Richard, A., Wegner, J. D. & Schindler, K. SUPERVISED OUTLIER DETECTION IN LARGE-SCALE MVS POINT CLOUDS FOR 3D CITY MODELING APPLICATIONS. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **IV–2**, 263–270 (2018).
10. Yan, L. *et al.* A New Outlier Removal Strategy Based on Reliability of Correspondence Graph for Fast Point Cloud Registration. *IEEE Trans. Pattern Anal. Mach. Intell.* 1–17 (2022) doi:10.1109/TPAMI.2022.3226498.
11. Yuan, X., Chen, H. & Liu, B. Point cloud clustering and outlier detection based on spatial neighbor connected region labeling. *Meas. Control* **54**, 835–844 (2021).
12. Rusu, R. B. & Cousins, S. 3D is here: Point Cloud Library (PCL). *IEEE Int. Conf. Robot. Autom.* (2011).

13. Cignoni, P. *et al.* 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 Ital. Chapter Conf.* (2008) doi:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.
14. Zhou, Q.-Y., Park, J. & Koltun, V. Open3D: A Modern Library for 3D Data Processing. (2018).
15. Zampogiannis, K., Fermüller, C. & Aloimonos, Y. cilantro: A Lean, Versatile, and Efficient Library for Point Cloud Data Processing. *ACM Int. Conf. Multimed.* (2018) doi:10.1145/3240508.3243655.
16. Rakotosaona, M. J., La Barbera, V., Guerrero, P., Mitra, N. J. & Ovsjanikov, M. PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds. *Comput. Graph. Forum* **39**, 185–203 (2020).
17. Diaz, N., Gallo, O., Caceres, J. & Porras, H. Real-time ground filtering algorithm of cloud points acquired using Terrestrial Laser Scanner (TLS). *Int. J. Appl. Earth Obs. Geoinformation* **105**, 102629 (2021).
18. Zhang, F., Zhang, C., Yang, H. & Zhao, L. Point cloud denoising with principal component analysis and a novel bilateral filter. *Trait. Signal* **36**, 393–398 (2019).
19. Wolff, K. *et al.* Point Cloud Noise and Outlier Removal for Image-Based 3D Reconstruction. (2016).
20. Rakotosaona, M.-J., Barbera, V. L., Guerrero, P., Mitra, N. J. & Ovsjanikov, M. PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds. *Comput. Graph. Forum* **xx**, 1–17 (2020).
21. Edl, M., Mizerák, M. & Trojan, J. 3D LASER SCANNERS: HISTORY AND APPLICATIONS. *Acta Simulatio-Int. Sci. J. Simul.* 4–5 (2018) doi:10.22306/asim.v4i4.54.
22. Spring, A. P. A History of Laser Scanning, Part 1: Space and Defense Applications. *Photogramm. Eng. Remote Sens.* **86**, 419–429 (2020).
23. Lewis, R. A. & Johnston, A. R. *A Scanning Laser Rangefinder for a Robotic Vehicle*. https://ntrs.nasa.gov/citations/19770015523 (1977).

24. Raj, T., Hashim, F. H., Huddin, A. B., Ibrahim, M. F. & Hussain, A. A Survey on LiDAR Scanning Mechanisms. *Electron. 2020 Vol 9 Page 741* **9**, 741–741 (2020).
25. Kinzel, P. J., Wright, C. W., Nelson, J. M. & Burman, A. R. Evaluation of an Experimental LiDAR for Surveying a Shallow, Braided, Sand-Bedded River. *J. Hydraul. Eng.* **133**, 838–842 (2007).
26. Riris, H. *et al.* The challenges of measuring methane from space with a lidar. in *International Conference on Space Optics — ICSO 2018* vol. 11180 829–838 (SPIE, 2019).
27. Göldner, D., Karakostis, F. A. & Falcucci, A. Practical and technical aspects for the 3D scanning of lithic artefacts using micro-computed tomography techniques and laser light scanners for subsequent geometric morphometric analysis. Introducing the StyroStone protocol. *PLOS ONE* **17**, e0267163 (2022).
28. Library of Congress. LAS (LASer) File Format, Version 1.4. *Digital Preservation* (2019).
29. Rusu, R. B. The PCD (Point Cloud Data) file format. *22.8.2020* (2020).
30. Library of Congress. Polygon File Format (PLY) Family. *Sustainability of Digital Formats: Planning for Library of Congress Collections* (2019).
31. Han, X. F. *et al.* A review of algorithms for filtering the 3D point cloud. *Signal Process. Image Commun.* **57**, 103–112 (2017).
32. Suchocki, C. & Błaszczak-Bąk, W. Down-Sampling of Point Clouds for the Technical Diagnostics of Buildings and Structures. *Geosciences* **9**, 70 (2019).
33. Zwicker, M. & Gotsman, C. Meshing Point Clouds Using Spherical Parameterization. *Eurographics Symp. Point-Based Graph.* (2004).
34. Rusu, R. B. Downsampling a PointCloud using a VoxelGrid filter. Documentation - Point Cloud Library (PCL). (2014).
35. Lu, D., Zhao, H., Jiang, M., Zhou, S. & Zhou, T. A Surface Reconstruction Method for Highly Noisy Point Clouds.

36. Schall, O., Belyaev, A. & Seidel, H.-P. Robust filtering of noisy scattered point data. in *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.* 71–144 (2005). doi:10.1109/PBG.2005.194067.

37. Digne, J. & De Franchis, C. The Bilateral Filter for Point Clouds. *Image Process. Line* **7**, 278–287 (2017).

38. Du, H., Yan, X., Wang, J., Xie, D. & Pu, S. Point Cloud Upsampling via Cascaded Refinement Network. *arXiv.org* https://arxiv.org/abs/2210.03942v1 (2022).

39. PU-GAT: Point cloud upsampling with graph attention network. *Graph. Models* **130**, 101201 (2023).

40. Liu, Z.-S., Wang, Z. & Jia, Z. Arbitrary point cloud upsampling via Dual Back-Projection Network. Preprint at https://doi.org/10.48550/arXiv.2307.08992 (2023).

41. Yao, Y., Fatholahi, S., Chen, Y. & Li, J. Indoor LiDAR Point Clouds Upsampling for Object Detection Enhancement. in *2023 Joint Urban Remote Sensing Event (JURSE)* 1–4 (2023). doi:10.1109/JURSE57346.2023.10144181.

42. Alexa, M. *et al.* Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph.* **9**, 3–15 (2003).

43. Yuan, C., Yu, X. & Luo, Z. 3D point cloud matching based on principal component analysis and iterative closest point algorithm. in *2016 International Conference on Audio, Language and Image Processing (ICALIP)* 404–408 (2016). doi:10.1109/ICALIP.2016.7846655.

44. Chen, X. & Chen, C.-H. Model-based point cloud alignment with principle component analysis for robot welding. in *2017 International Conference on Advanced Robotics and Intelligent Systems (ARIS)* 83–87 (2017). doi:10.1109/ARIS.2017.8297194.

45. Cignoni, P. *et al.* 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 Ital. Chapter Conf.* (2008) doi:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

46. Panozzo, D. & Jacobson, A. libigl: Prototyping Geometry Processing Research in C++. *EUROGRAPHICS* (2019) doi:10.2312/egt.20191037.

47. Zhou, Q.-Y., Park, J. & Koltun, V. Open3D: A Modern Library for 3D Data Processing. (2018).

48. Point cloud outlier removal - Open3D primary (0cf605f) documentation. https://www.open3d.org/docs/latest/tutorial/geometry/pointcloud_outlier_removal.html.

49. Griffiths, D. & Boehm, J. A review on deep learning techniques for 3D sensed data classification. *arXiv.org* https://arxiv.org/abs/1907.04444v1 (2019) doi:10.3390/rs11121499.

50. Qiu, S., Anwar, S. & Barnes, N. Geometric Back-projection Network for Point Cloud Classification. *arXiv.org* https://arxiv.org/abs/1911.12885v5 (2019).

51. Pan, L. *et al. Variational Relational Point Completion Network*. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2021) doi:10.1109/CVPR46437.2021.00842.

52. Wang, X., Fan, X. & Zhao, D. PointFilterNet: A Filtering Network for Point Cloud Denoising. *IEEE Trans. Circuits Syst. Video Technol.* **33**, 1276–1290 (2023).

53. Mémoli, F. & Sapiro, G. Comparing point clouds. in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* 32–40 (ACM, Nice France, 2004). doi:10.1145/1057432.1057436.

54. Urbach, D., Ben-Shabat, Y. & Lindenbaum, M. DPDist : Comparing Point Clouds Using Deep Point Cloud Distance. *arXiv.org* https://arxiv.org/abs/2004.11784v2 (2020) doi:10.1007/978-3-030-58621-8_32.

55. Barrow, H. G. PARAMETRIC CORRESPONDENCE AND CHAMFER MATCHING: TWO NEW TECHNIQUES FOR IMAGE MATCHING. (1977).

56. Pan, L. *et al. Variational Relational Point Completion Network*. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 8520–8529 https://paul007pl.github.io/projects/VRCNet (2021) doi:10.1109/CVPR46437.2021.00842.

57. Huang, T., Liu, Q., Zhao, X., Chen, J. & Liu, Y. Learnable Chamfer Distance for Point Cloud Reconstruction. Preprint at http://arxiv.org/abs/2312.16582 (2023).

58. Pedegrosa et al. scikit-learn Machine Learning in Python. *JMLR 12* 2825–2830 https://scikit-learn.org/stable/ (2011).

59. Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 90–95 https://matplotlib.org/ (2007).

60. Daghigh, H., Tannant, D. D., Daghigh, V., Lichti, D. D. & Lindenbergh, R. A critical review of discontinuity plane extraction from 3D point cloud data of rock mass surfaces. *Comput. Geosci*. **169**, 105241 (2022).