Novel algorithms for 3D surface point cloud boundary detection and edge reconstruction

Carmelo Mineo*, Stephen Gareth Pierce, Rahul Summan

Department of Electronic and Electrical Engineering, University of Strathclyde
Royal College Building, 204 George Street, Glasgow G1 1XW, UK

* Corresponding author: *carmelo.mineo@strath.ac.uk*

Abstract

Tessellated surfaces generated from point clouds typically show inaccurate and jagged boundaries. This can lead to tolerance errors and problems such as machine judder if the model is used for ongoing manufacturing applications. This paper introduces a novel boundary point detection algorithm and spatial FFT-based filtering approach, which together allow for direct generation of low noise tessellated surfaces from point cloud data, which are not based on pre-defined threshold values. Existing detection techniques are optimized to detect points belonging to sharp edges and creases. The new algorithm is targeted to the detection of boundary points and it is able to do this better than the existing methods. The FFT-based edge reconstruction eliminates the problem of defining a specific polynomial function order for optimum polynomial curve fitting. The algorithms were tested to analyse the results and measure the execution time for point clouds generated from laser scanned measurements on a turbofan engine turbine blade with varying numbers of member points. The reconstructed edges fit the boundary points with an improvement factor of 4.7 over a standard polynomial fitting approach. Furthermore, through adding artificial noise it has been demonstrated that the detection algorithm is very robust for out-of-plane noise lower than 25% of the cloud resolution and it can produce satisfactory results when the noise is lower than 75%.

Keywords
Point-cloud, boundary detection, edge reconstruction

1. Introduction

Three-dimensional (3D) scanning is increasingly used to analyse objects or environments in diverse applications including industrial design, orthotics and prosthetics, gaming and film production, reverse engineering and prototyping, quality control and documentation of cultural and architectural artefacts [1, 2]. Conventional reconstruction techniques generate tessellated surfaces from point clouds. Such tessellated models often show inaccurate and jagged boundaries that can lead to tolerance errors and problems such as machine judder if the models are used for ongoing manufacturing applications [3]. That is the reason why many existing commercial computer-aided manufacturing (CAM) applications are not able to use tessellated models, instead using precise analytical CAD models where the surfaces are mathematically represented [4]. Whilst the conversion of analytical geometries into meshed surfaces is straightforward, the reverse process of conversion of a tessellated model into an analytical CAD model is challenging and time-consuming. There are circumstances where the original CAD model of a component is not available or deviates from the real part. New CAM software applications, able to use clean tessellated models, are emerging [4]; they enable the direct use of triangulated point clouds obtainable through surface mapping techniques. However, the point clouds obtained through surface mapping are typically affected by noise. New algorithms for optimum surface mesh refinement are required to improve the performance of emerging applications and to overcome the limitations of typical approaches based on polynomial smoothing. Different technologies can be used to build coordinate measuring machines (CMM) or 3D-scanning devices [1]. Each technology comes with its own limitations, advantages and costs. A common factor for many CMM and 3D scanners is that they can measure the coordinates of a large number of points on an object surface and output a point cloud of the scanned area. . However, point clouds are generally not directly usable in most 3D applications, and therefore are usually converted to mesh models, NURBS surface models, or CAD models [5-7]. Tessellated models have emerged as a favoured technique; they are the easiest form of virtual models obtainable from point clouds with minimal processing. There are two different approaches to create a triangular meshed surface from a point cloud: using triangulation methods or surface reconstruction methods. Triangulation algorithms use the original points of the input point cloud, using them as the vertices of the mesh triangles. The Delaunay triangulation, named after Boris Delaunay for his work on the topic from 1934 [8], is the most popular algorithm of this kind. A bi-dimensional Delaunay triangulation ensures that the circumcircle associated with each triangle contains no other point in its interior. This definition extends naturally to three dimensions considering spheres instead of circles. Surface reconstruction algorithms differ from the triangulation method since they do not use the original points as the vertices of the mesh triangles but compute new points, whose density can vary according to the local curvature of the 3D geometry. Surface reconstruction from oriented points can be cast as a spatial problem, based on the Poisson's equation [9, 10].

Both approaches are not able to reconstruct the surface boundaries accurately, which makes the tessellated models unsuitable to be used for CAM toolpath generation. Triangulation methods produce meshed surfaces with jagged boundaries, since the original noisy points of the cloud are used as vertices of the mesh triangles. Reconstruction methods produce smooth boundaries, but they can be quite far from the original boundaries of the real surface. Indeed, Poisson’s surface reconstruction does not follow the boundary of the point cloud and replaces the original points with new points laying on a reconstructed continuous surface that satisfies the Poisson’s differential equation.

The detection of cloud boundary points and the reconstruction of smooth boundary edges would allow trimming of the reconstructed tessellated models, to refine the mesh boundaries. Very few feature detection methods are optimized to work with point-sampled geometries only. The major problem of these point based methods is the lack of knowledge concerning point normal and connectivity. This makes feature detection a more challenging task than in mesh based methods. Gumhold et al. [11] presented an algorithm that first analyses the neighbourhood of each point via a principal component analysis (PCA). The eigenvalues of the correlation matrix are then used to determine if a point belongs to a feature. This technique for the detection of features in point clouds is used as a pre-processing step for tessellated surface reconstruction with sharp features

[12]. There exist also several reconstruction methods that preserve sharp features during the surface reconstruction of a point cloud without pre-processing; for example the methods shown by Fleischmann et al. [13] and Öztireli et al. [14].

The existing techniques mentioned above are optimized to detect points belonging to sharp edges. This paper presents novel algorithms targeted to the detection of boundary points and the deterministic reconstruction of accurate and smooth surface boundaries from 3D point clouds. A smart approach known as Mesh Following Technique (MFT) [4], for the generation of robot tool-paths from STL models, has recently been published. The technique requires virtual tessellated surfaces with smooth boundary edges.

The algorithms presented in this paper are useful tools to refine the boundary of tessellated surfaces obtained from 3D scanning point cloud data. They can be used to trim Delaunay triangulation or Poisson's reconstructed surface meshes, facilitating the direct use of tessellated models, instead of analytical geometries. The remainder part of the paper describes the algorithms and shows qualitative and quantitative results, discussing advantages and disadvantages.

2. Detection of boundary points

Given a mapped surface in the form of a point cloud, the identification of the point cloud borderline, thus the detection of boundary points, is not a trivial task. The human brain is able to infer the border of a point cloud by simply looking at the arrangement of the sparse points. In computer science and computational geometry, a point cloud is an entity without a well-defined boundary. In the bi-dimensional domain, given a finite set of points, the problem of detecting the smallest convex polygon that contains all the given points of the cloud is solved through the *quickhull* method [15]. It uses a divide and conquer approach. This method works well but is only able to detect the boundary points that are part of the convex polygon and is only for bi-dimensional set of points. A generalization of *quickhull*, able to handle concave regions and holes in the point cloud, is the *alpha-shape* approach [16]. The Computational Geometry Algorithms Library (CGAL) [17] has a robust implementation of *alpha-shape* for 2D and 3D point clouds. For each real number α, the approach is based on the generalized disk of radius $1/\alpha$. An edge of the polygon that contains all the given points (alpha-shape) is drawn between two members of the finite point set whenever there exists a generalized disk of radius $1/\alpha$ containing the entire point set and which has the property that the two points lie on its boundary. If $\alpha = 0$, then the alpha-shape associated with the finite point set is its ordinary convex hull given by *quickhull*. The limitation of the *alpha-shape* approach is that its performance depends on the set value of the parameter α. A value of α that produces a satisfactory result for a point cloud may not be suitable for other point sets, since point clouds can exhibit different point densities. This inconvenience is similar to what happens when obtaining a black and white picture from a grayscale image, through thresholding the pixel intensities; the optimal threshold value is affected by the average brightness of the image. Moreover, when the point density of a point cloud varies between across the cloud, the alpha-shape result can be satisfactory in some regions and poor in others. Non-parametric edge extraction methods based on kernel regression [14] and on analysis of eigenvalues [18] have been proposed in recent years. Such methods, however, are optimized for the detection of internal sharp edges, rather than detecting the point cloud borderline.

The boundary point detection algorithm presented in this paper, herein referred as BPD algorithm, does not need the definition of any threshold values. For every region of the cloud, it detects as many boundary points as possible, given the local resolution of the region. A 3D-point cloud is unorganized and the neighbourhood of a point is more complex than that of a pixel in an image. Generally, in 3D-point clouds, there are three types of neighbourhoods: spherical neighbourhood, cylindrical neighbourhood, and k-nearest neighbours based neighbourhood [19]. The three types of neighbourhoods are based on different search methods. Given a point P, a spherical neighbourhood is formed by all 3D points in a sphere of fixed radius around P. A cylindrical neighbourhood is formed by all those 3D points whose 2D projections onto a plane (e.g. the ground plane) are within a circle of fixed radius around the projection of P. The k-nearest neighbourhood (k-NN) search method is non-parametric and it is used in this work, since it does not need the definition of a radius value; it finds the closest k-members of the cloud. Figure 1 shows a point cloud with the k-nearest neighbourhood of 5 points (A to E), where k is set to 30. The points of the cloud belonging to the neighbourhoods are indicated through filled circles. The other points of the cloud are represented with empty circles. The semi-transparent circles, centred at the points from A to E, highlight the best fit planes for each of the neighbourhoods. The normal directions of such planes are also shown through arrows pointing outwards from the neighbourhood parent points. The local cloud resolution for every member of the cloud is estimated through the following steps. Given a point of the cloud P_i, for every point of its neighbourhood ($P_{i,j}$), the minimum distance ($d_{j,k}$) between that point and all other neighbours is computed. The local point cloud resolution (β_i) in P_i is estimated as $\beta_i = \mu_i + 2\sigma_i$, where μ_i is the mean value of the minimum distances and σ_i is their standard deviation. This method of computing the local resolution is robust, overcoming the problematic noisy nature of some point clouds collected through optical and photogrammetric method. If the distance values ($d_{j,k}$) are distributed according to a Gaussian distribution, the addition of $2\sigma_i$ to the mean value ensures that 97.6% of the data values are considered and the major outliers are ignored.

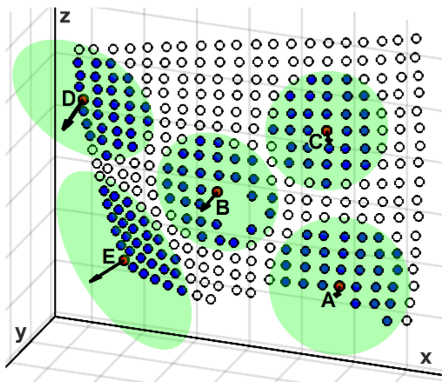

Figure 1 – Point cloud with examples of two boundary points on concave regions (A and B), an inner point (C) and two boundary points on convex regions (D and E).

The alpha-shape method may not be able to detect some sections of the point cloud boundary with high concave curvature, if α is set too low. On the other hand, it may detect unwanted boundary points, if α is too high. The new method described here is capable of detecting boundary points belonging to convex and concave regions.

The BPD method exploits the fact that there is one and only one circle that passes through three given points in the 3D space. Given the unclassified point P_i, the centre and the radius of all circles that pass for P_i and any two other points of its neighbourhood are computed. The point is labelled as a boundary point if there is at least one circle with radius equal or bigger that β_i and if the sphere for P_i, whose centre coincide with the centre of the circle, does not contain any other point of the neighbourhood. This is the case for point A and B, shown in Figure 1. Point A and B are two boundary points, since it is possible to find at least one circle that satisfies the above conditions (Figure 2). In order to avoid unnecessary computational efforts and yet consider all possible circles, all the originating point triples are identified through the binomial coefficient [20]. Working with k-nearest neighbourhoods and being the investigated point P_i always part of the triples, the remaining points (k-1) are combined in couples with no repetitions. Thus, the total number of circles is equal to:

$$n = \frac{(k-1)!}{2! \cdot [(k-1)-2]!} = \frac{29!}{2 \cdot 27!} = \frac{29 \cdot 28}{2} = 406 \quad (1)$$

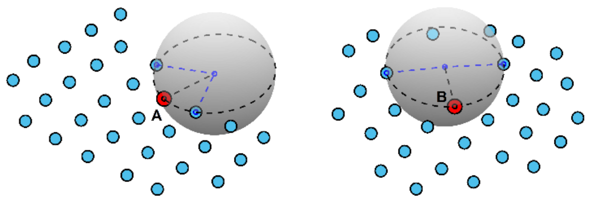

Figure 2 – A and B labelled as boundary points.

Although the presence of at least one circle that satisfies the above conditions allows to classify the investigated point as belonging to the point cloud boundary, its absence cannot be used to state that the point is an internal point of the cloud. Indeed the investigated point P_i can be located on a convex region of the boundary as well as being an internal point of the cloud. In such circumstances, although it may exist one circle with radius equal or bigger that β_i, the sphere for P_i, centred at the centre of the circle, will always contain some points of the neighbourhood. This is the case for the points C, D and E shown in in Figure 1. For such kind of points, thus when the point cannot be labelled as a boundary point through the first part of the detection algorithm described above, the algorithm continues with further operations. Each investigated point and its neighbours are projected to the best fit plane according to the normal vector associated with the point. The resulting bi-dimensional neighbourhood cloud can be plotted in polar coordinates, with P_i at the pole of the plot. Figure 3 shows the polar plots for point C, D and E and their neighbours. P_i is shown in red and its neighbour points are shown in blue. The blue and red dotted line of the plots in Figure 3 highlight, respectively, the minimum and maximum angle of the angular sector spanned by the points in the neighbourhood.

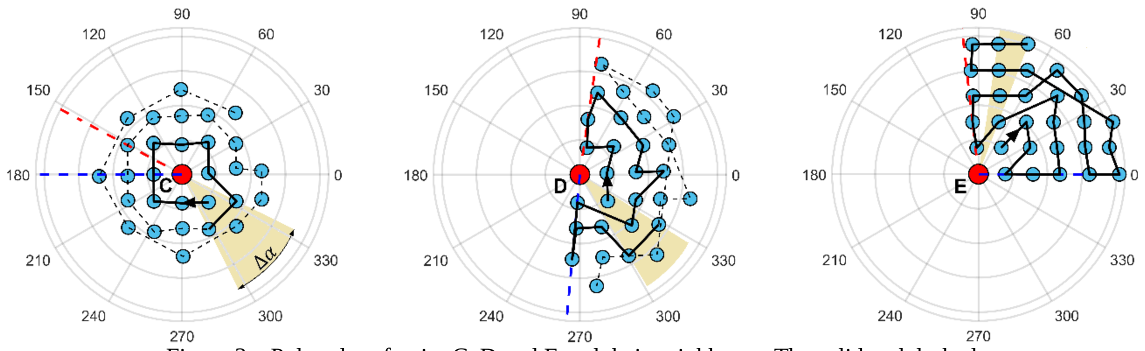

Figure 3 – Polar plot of point C, D and E and their neighbours. The solid and dashed lines illustrate the application of the algorithm.

The fundamental idea behind the final step of the BPD algorithm is that the point of the cloud P_i is a boundary point if it is not possible to find a path that surrounds it and passes through the neighbour points. Each point on the polar plot is determined by the distance from the pole (radial coordinate, R) and the angular coordinate (θ). Given a neighbourhood, the developed algorithm creates a path that surrounds the parent point at the pole. An incremental approach is used. All radial and angular coordinates of the neighbours are normalized, so that the coordinates of the j-th neighbour point are:

$$r_j = \frac{R_j - R_{min}}{R_{max} - R_{min}} \tag{2}$$

$$\vartheta_j = \frac{\theta_j - \theta_{min}}{\theta_{max} - \theta_{min}} \tag{3}$$

where R_{min}, R_{max} and θ_{min}, θ_{max} are respectively the minimum and maximum values of the radial and angular coordinates. In order to select the starting point of the surrounding path, a characterizing parameter (γ) is given to each neighbour, such that:

$$\gamma_j = r_j + \left|\vartheta_j - \vartheta_{bis}\right| \tag{4}$$

where ϑ_{bis} is the normalized angle of the direction bisecting the angular gap comprising all neighbours (between θ_{min} and θ_{max}). The neighbour point with the minimum value of γ, as computed in Eq. 4, is selected as starting point for the surrounding path. From the starting point the algorithm progresses through linking the other points of the neighbourhood. The crossed neighbours are removed from the list of available points. The characterizing parameter of the generic p-th remaining point is computed as:

$$\gamma_p = r_p + \left[\left(\vartheta_p - \vartheta_{last}\right) \cdot c\right] \tag{5}$$

where ϑ_{last} is the normalized angular coordinate of the last crossed point and c is a factor equal to -1 if $sgn(\vartheta_p - \vartheta_{last}) = sgn(\vartheta_{last} - \vartheta_{last-1})$ or equal to 1 otherwise. The factor c facilitates the selection of a point that does not force a change of surrounding direction. For example, if the last selected point produced a clockwise rotation, any anti-clockwise rotation is penalised. The neighbour with the minimum value of γ, as computed in Eq. 5, becomes the new last point of the path.

The incremental linkage of the neighbours keeps track of the spanned angle around the pole. The sum of the angle increments considered with their sign ($\alpha = \sum \Delta\alpha$) and the sum of their absolute values ($\tau = \sum|\Delta\alpha|$) are updated each time a new point is selected after the starting point. The incremental linkage stops when there are no more linkable neighbour points or when $\tau > 2\pi$. If $|\alpha| \geq 2\pi$ the investigated point is an internal point of the cloud (e.g. point C in Figure 3), otherwise it is a boundary point (e.g. D and E in

Figure 3). The stopping condition of the incremental algorithm ($\tau > 2\pi$) avoids superfluous computation efforts, since it is often possible to infer if a point belongs to the boundary without linking all neighbours. The solid line in Figure 3 shows the surrounding path created until $\tau > 2\pi$. For example, it is possible to determine that C does not belong to the boundary, linking only 10 out of 29 neighbours, since $|\alpha| > 2\pi$. It is possible to state that D is a boundary point, through linking 18 out of 29 neighbours; it results $|\alpha| < 2\pi$, although the sum of the absolute angle increments is $\tau > 2\pi$. In summary, the BPD method described here is capable of determining if a point P_i belongs to a concave boundary sections (e.g. point A in Figure 1), when the local curvature is as high as $1/\beta_i$. The method is also able to infer if a point belongs to the boundary of a hole in the point cloud (e.g. point B in Figure 1), when the hole radius is as small as β_i. Since β_i is the local resolution, the method works well on point clouds with variable point density.

3. Edge reconstruction

The application of the BPD algorithm to the point cloud sample shown in Figure 1 finds all the boundary points, represented as empty circles in Figure 4. Also the 4 boundary points of the internal circular hole, whose radius is just above the local point cloud resolution, are detected as expected. The detected points need to be clustered, so that points belonging to the same boundary are grouped together. Moreover, the points of every cluster need to be ordered correctly. These tasks can be fulfilled through existing algorithms. For example, the clustered points can be ordered through algorithms capable of solving the so called *travelling salesman problem* (TSP). Given a random list of points and the distances between each pair of points, the solution of the TSP is the shortest possible path that crosses each point exactly once and returns to the origin point [21]. Therefore a closed boundary path is obtained from every cluster. These boundaries, given by the ordered points linked through line segments (see dashed lines in Figure 4) can be quite jagged in some areas. Therefore, it is evident that such boundaries are not suitable to trim the surface meshes obtainable from the point cloud. The boundary curves need smoothing to better resemble the real surface borderlines. This section of the paper introduces a novel raw boundary smoothing algorithm, herein referred as RBS algorithm, to improve the reconstruction of surface point cloud borderlines through accurate smoothing of the raw boundaries.

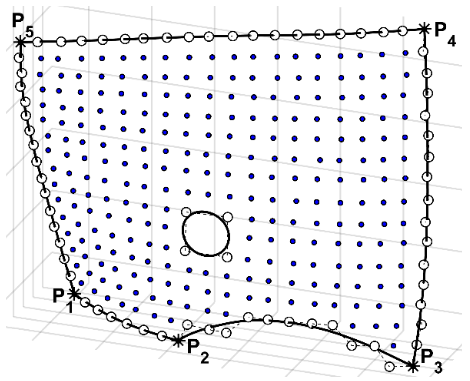

Figure 4 – Closed boundaries partitioned into edges, obtained through clustering and ordering of the detected boundary points (jagged dashed line) and reconstructed edges (solid lines).

3.1 Detection of key points

Before applying any smoothing algorithm to the closed loop boundary curves, it is necessary to highlight that the position of some of the detected boundary points should be preserved. This is the case for the borderline corners, where there is a sharp change in the boundary directionality. Indeed, such points usually play a crucial role in the definition of reference systems in CAM applications for the development of accurate operations, where the correct registration of the part virtual model is required. Therefore, the first step of the edge reconstruction algorithm is targeted to detect such key points. Given the i-th point of a closed boundary path, the point is labelled as corner if the radius of the circle for the investigated point, the precedent and the successive point is smaller than the value of the local point cloud resolution, β_i (as calculated in the detection algorithm). This approach is able to identify the corner points from P_1 to P_5, highlighted with asterisks in Figure 4. It is worth noticing here that, although the corners can be found through applying a threshold value to the angle between the two borderline segments for each investigated point [22], this approach is advantageous and it is suitable to work with point clouds with variable density. The identified corners are used to divide the closed loop boundary into sections, corresponding to the edges of the surface geometry. The external boundary points of the cloud in Figure 4 are grouped in 5 edges. The 4 internal boundary points are grouped in one single closed loop edge, since no corners are found there.

3.2 Limitation of traditional curve smoothing methods

Each edge could be smoothed through fitting a polynomial curve. Polynomial curve fitting is a common smoothing method and the functionality is also implemented in CAD/CAM commercial software applications (e.g. Rhinoceros®). Curve fitting is the process of approximating a pattern of points with a mathematical function [23]. Fitted curves can be used to infer values of a function where no data are available [24] (e.g. in the gaps between sampled points). The goal of curve fitting is to model the expected value of a dependent variable y in terms of the value of an independent variable (or vector of independent variables) x. In general, the expected value of y can be modelled as an nth degree polynomial function, yielding the general polynomial regression model based on the truncated Taylor's series:

$$y = a_0 + a_1x + a_2x^2 + \dots + a_nx^n + \varepsilon \qquad (6)$$

where ε is a random error with null mean. Given the points of an edge and the order of the target polynomial function, it is possible to compute the coefficients of the fitting function. The limitation of this smoothing approach is that the order of the target function is typically unknown and curve fitting remains a time-consuming iterative trial and error process for edge reconstruction. When there is no theoretical basis for choosing the order of the fitting polynomial function, the edges may be fitted with a spline functions composed of a sum of B-splines [25]. The places where the B-splines meet are known as knots. The main difficulty in applying this process is in determining the number of knots to use and where they should be placed [26].

3.3 FFT-based reconstruction

The final step of the RBS algorithm, presented in this paper, introduces a robust approach based on the Fast Fourier Transform (FFT). The FFT is a well-known way to translate the information contained in a waveform from the time domain to frequency domain. It is used for the spectral analysis of time-series and allows the application of high or low-pass filters, to respectively attenuate low or high frequencies. Here the FFT is applied to the pattern of the edge point Cartesian coordinates, to enable the application of low-pass filters able to improve the smoothness of the boundary edges. The exploitation of FFT to spatial patterns (waveforms sampled in the Cartesian domain rather than the time domain) is not new (e.g. it has been used for image processing [27]). However, there is no record of the FFT being applied to the problem of surface edge reconstruction. The nuances of the adaptation of FFT to this problem are described herein.

Consider a series $x(k)$ with N samples. Furthermore, assume that the series outside the range between 0 and N-1 is extended N-periodic, which is $x(k) = x(k + N)$ for all k. The FFT of this series will be denoted $X(k)$, it will also have N samples. The FFT transform implies specific relationships between the series index and the frequency domain sample index. For the common case, where the FFT is applied to series representing a time sequence of length T, the samples in the frequency domain are spaced by $f_s = 1/T$. The first sample $X(0)$ of the transformed series is the average of the input series. The frequency sample corresponding to $f_{Ny} = N/2T$ is called Nyquist frequency. This is the highest frequency component that should exist in the input series for the FFT to yield uncorrupted results. More specifically if there are no frequencies above Nyquist the original pattern can be exactly reconstructed from the samples in the frequency domain. For the spatial problem of edge reconstruction, given the FFT is applied to the edge Cartesian component pattern, plotted as function of the curvilinear distance (d), the spatial frequency is a measure of how often sinusoidal components (as determined by the FFT) repeat per unit of distance. The spatial frequency domain representation of any Cartesian component of a circular edge with radius (R) and length ($2\pi R$) contains only one frequency component $f_c = 1/2\pi R$. Therefore it is possible to deduce that, denoting the local curvature radius at the i-th point of the edge with R_i, the main spatial frequency occurring at that point is equal to $f_i = 1/2\pi R_i$. According to the Nyquist theorem, when sampling an analogue signal in the time domain, the sampling rate must be at least equal to $2f_{max}$, where f_{max} is the highest frequency component. The Nyquist rule applied to the spatial domain means that β_i (the local point cloud resolution) limits the minimum edge radius that is possible to reconstruct at the i-th point. The smallest radius that is possible to reconstruct will be the one associated to a circumference of length $2\pi\beta_i$ sampled with 2 points, corresponding to the spatial frequency $f^* = 2/2\pi\beta_i = 1/\pi\beta_i$. The maximum alias-free spatial frequency component will be:

$$f_{max} = f^*/2 = 1/2\pi\beta_i. \tag{7}$$

The smallest edge radius of curvature that is possible to reconstruct at the i-th point will be equal to $R_i^{max} = \beta_i$.

The plots in Figure 5a and 5b regard, respectively, the x-component pattern of the edge between P_2 and P_3 and of the closed loop internal hole edge of the cloud in Figure 4. The patterns are plotted as functions of the normalized curvilinear distance of the edge ($d^* = d/D$, with D being the total length of the edge). The original patterns, given by the dashed line that goes through the x-component samples (shown through round circles), are quite jagged. Figure 5 clarifies how a periodic waveform is obtained from the original pattern of each Cartesian component of a given edge. The pattern is first translated along the direction of the ordinate axis to move the first point of the pattern to the origin of the plot. The pattern is then rotated by the angle α to move the last point of the pattern on the horizontal axis. A copy of the resulting pattern is inverted, flipped and appended to the end extremity; it constitutes a complementary portion creating a period with the translated and rotated version of the original pattern (Figure 5a). Since the FFT assumes a constant sampling rate of the input pattern, the original randomly spaced samples are replaced with interpolated equally spaced samples. The number of interpolated samples (N_p) is chosen appropriately to give a constant sampling interval (d_s), equal or smaller than the minimum original sampling distance. In order to ensure a good filtering performance, it is necessary to have sufficient spatial frequency resolution. For such reason the period is repeated to get a minimum of 1000 samples in the input waveform of the FFT, giving a frequency resolution of $f_s = 1/(1000 \cdot d_s)$. Therefore a low-pass filter is applied, cancelling all spatial frequency components higher than f_{max}, as expressed in Eq. 7. This produces a smoothed waveform for the edge component. The FFT input waveform shown in Figure 5a, artificially constructed to filter the x-component of the open edge comprised between P_2 and P_3, has a null mean value. The original edge extremities lie on the horizontal axis (the mean value line) and are not affected by the low-pass filtering. The Cartesian component values of the extremities are preserved. The first part of the waveform (the portion up to the total length D of the original pattern) is rotated by negative α and translated back to the original position. The smoothed boundary is obtained through filtering all the Cartesian components of all its edges. The preservation of the original edge extremity points makes sure that, when a boundary consists of multiple edges, two consecutive edges share a common point. Therefore, the chain of edges forms a closed boundary. If a boundary is formed by only one closed edge, like in the case of the internal hole boundary in Figure 4, the extremities of each of its Cartesian components have the same value ($\alpha = 0$). Moreover the complementary portion to construct the period is a mere horizontally translated copy of the Cartesian component pattern (after its extremities are brought to the horizontal axis). The copy of the original pattern is not inverted nor flipped to create the complementary portion (Figure 5b). All points of the closed edge are affected by the filtering. The smoothed edges relative to the boundaries of the sample point cloud are shown through solid line curves in Figure 4.

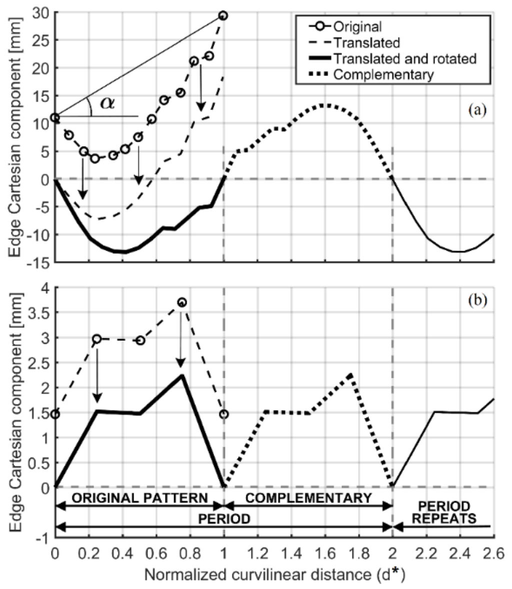

Figure 5 – Creation of periodic waveform for application of spatial FFT to the x-component pattern of an open edge (a) and a closed edge (b).

4. Results and performances

This section of the paper analyses the results obtainable through the use of the introduced BPD and RBS algorithms. The computational performances are also examined and quantitative figures are reported. Figure 6 shows a schematic summary of the algorithm steps. The thick dashed line perimeters contain the novel algorithm components introduced by this paper. The BPD algorithm (Figure 6a) allows the unlabelled point of a surface point cloud to be grouped into two groups: boundary points and internal points. The boundary points are clustered and ordered, through existing algorithms, to constitute raw closed boundaries (jagged). The RBS algorithm (Figure 6b) identifies the boundary corners and divides each closed boundary into the constituting edges. Every edge is smoothed through spatial FFT-based filtering. A crucial advantage of the introduced algorithms is that they are not based on any threshold values that can be suitable for some point cloud but not suitable for others. The BPD is capable of labelling all points, observing the local resolution of the cloud for each point. The FFT-based edge reconstruction eliminates the problem of defining a specific polynomial function order for optimum polynomial curve fitting. In the approach introduced in this paper, the best edge smoothing performance is also ensured through applying a spatial low-pass filtering with cut-off frequency defined at every boundary point as a function of the local cloud resolution.

In order to show the potentialities of the new algorithms, an aircraft turbine engine fan blade, 640 mm long and 300 mm wide (in average) was scanned through a coordinate measuring machine. The FARO Quantum Arm was used in conjunction with a laser profile mapping probe [28]. This 3D scanning equipment has a volumetric maximum deviation of ±74μm. The blade surface was scanned to obtain a uniform point cloud with circa 14 thousand points (approximately 72 points per square centimetre).

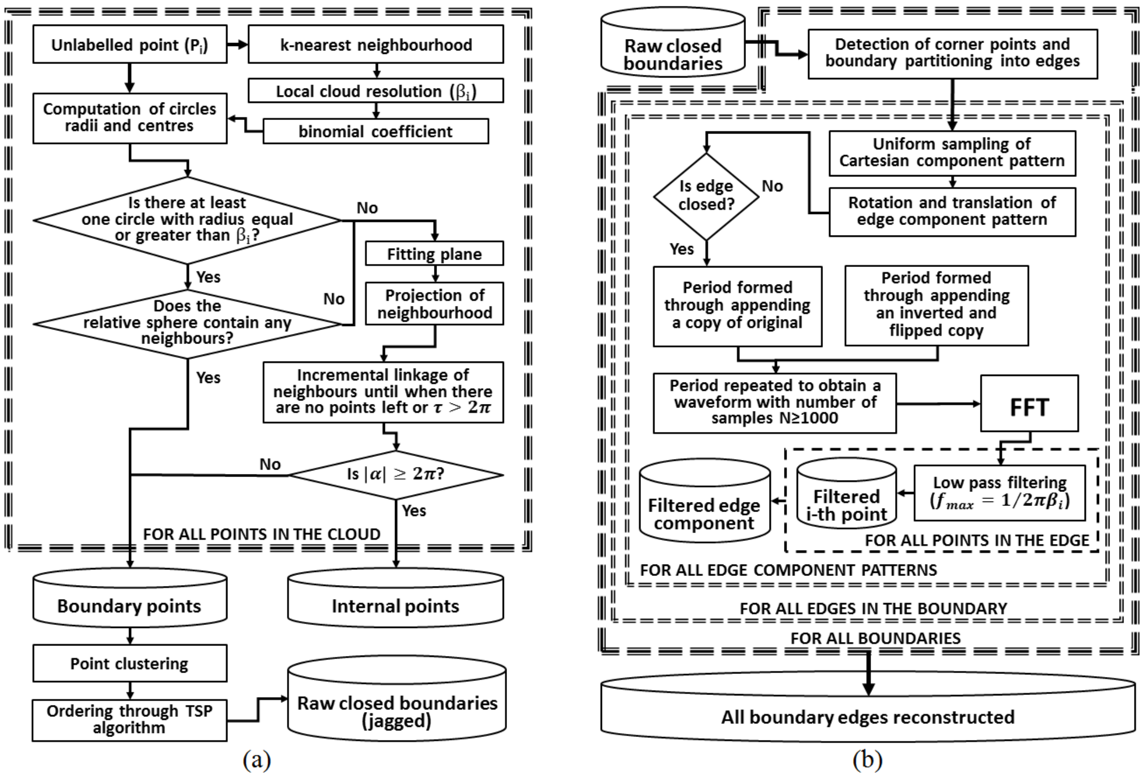
Unlabelled point (P_i)
k-nearest neighbourhood
Local cloud resolution (β_i)
Computation of circles radii and centres
binomial coefficient
Is there at least one circle with radius equal or greater than β_i?
No
Fitting plane
Projection of neighbourhood
Yes
Does the relative sphere contain any neighbours?
No
Yes
Incremental linkage of neighbours until when there are no points left or $\tau > 2\pi$
No
Is $|\alpha| \geq 2\pi$?
Yes
FOR ALL POINTS IN THE CLOUD
Boundary points
Internal points
Point clustering
Ordering through TSP algorithm
Raw closed boundaries (jagged)
(a)
Raw closed boundaries
Detection of corner points and boundary partitioning into edges
Uniform sampling of Cartesian component pattern
Is edge closed?
No
Rotation and translation of edge component pattern
Yes
Period formed through appending a copy of original
Period formed through appending an inverted and flipped copy
Period repeated to obtain a waveform with number of samples N≥1000
FFT
Filtered edge component
Filtered i-th point
Low pass filtering ($f_{max} = 1/2\pi\beta_i$)
FOR ALL POINTS IN THE EDGE
FOR ALL EDGE COMPONENT PATTERNS
FOR ALL EDGES IN THE BOUNDARY
FOR ALL BOUNDARIES
All boundary edges reconstructed
(b)

Figure 6 – Schematic summary of the algorithm steps: boundary point detection (a) and edge reconstruction (b).

The cloud points were decimated to obtain four different point cloud versions, with target resolution respectively equal to 4, 8, 16 and 34mm. An additional point cloud was generated with variable point resolution, between 2 and 34mm. Such generated point clouds allowed testing the algorithms under controlled situations and facilitated the analysis of the results. In order to introduce well-defined internal boundaries, the points found within three spheres centred at fixed positions and with radius equal to 16, 32 and 64mm were removed from the clouds. Therefore, each cloud presents three holes (H1, H2 and H3), with radii approximatively equal to the original generating spheres. The resulting five point clouds are shown by the top row plots in Figure 7. These plots show the detected boundary points through darker round point marks. The external and internal boundary points are detected as expected. The smallest radius of the hole detectable in a point cloud depends on the cloud resolution, as described in Section 2. Only four points of the 16mm and 32mm radius holes (H1 and H2) are detected in the clouds of Figure 7d and 7e, since the resolution of these clouds is close to their radii. H1 cannot be detected on the 34mm resolution cloud (Figure 7e).

Figure 8 shows the boundary points detected in the Stanford bunny [29], an open source computer graphics test meshed model, obtained through range scanning [30]. Similarly to how many researchers have used this model, as input for surface reconstruction algorithms, the mesh connectivity has been stripped away and the vertices have been treated as an unorganized point cloud in this work. It presents 35947 points and has an average resolution of 1.2mm. Boundary points were detected using a method based on principal component analysis (PCA) [11] (Figure 8a) and the new detection algorithm (Figure 8b). To appreciate the differences in performance it is important to differentiate boundary points and edge points; the latter points are located in areas where there is a sharp ripple (like in the bunny ears). The PCA method detected points that are either on sharp crease lines or on the borderline of the point cloud holes. The BPD algorithm is targeted to exclusively find borderline points; therefore it found two holes in the base of the bunny, originating from the original clay model (it was a hollow model). It should be noted that the BPD algorithm should not detect the extra edge and crease points detected by PCA (e.g. in the ears of the bunny). It is designed to detect boundary points (around holes or areas where the point density drops drastically), so that a boundary edge can be reconstructed and the tessellated surface can be trimmed. The BPD algorithm was capable of detecting four additional groups of boundary points (one group under the bunny chest, one group between the bunny front legs and two groups on the base), corresponding to the borderline of areas with poor range scanning coverage. The smallest two of these areas were not detected by the PCA method, showing how the BPD algorithm is more accurate for the detection of borderline points.

The bottom row plots of Figure 7 shows the reconstructed boundary edges. The effectiveness of the FFT-based filtering algorithm is evident observing the smoothness of the edges. The smoothed boundaries of the internal holes, given by single closed edges, faithfully reproduces the roundness of the theoretical intersection between the original generating sphere and the blade surface. Only the boundaries of H1 and H2, reconstructed through only four detected points, show visible distortion. Although explaining how mesh trimming works is out of the scope of this paper, the reconstructed boundary edges can be used to trim the Poisson mesh, producing the clean boundary meshes highlighted in Figure 7.

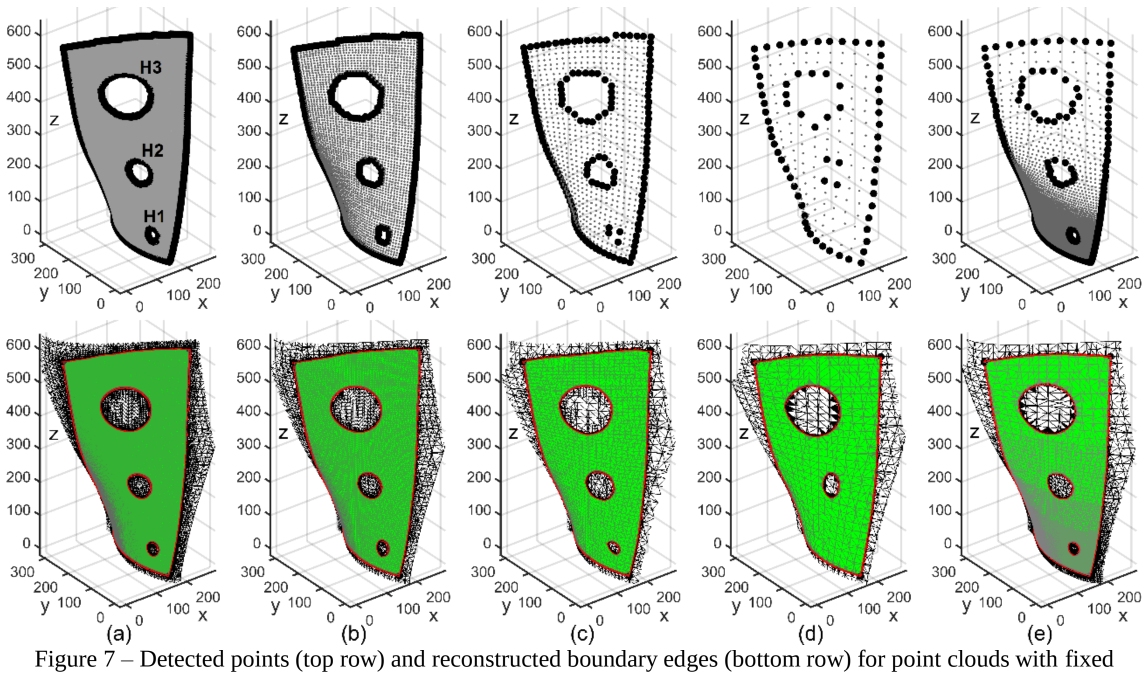

Figure 7 – Detected points (top row) and reconstructed boundary edges (bottom row) for point clouds with fixed resolution of 4 (a), 8 (b), 16 (c) and 34mm (d) and with variable resolution between 2 and 34mm (e).

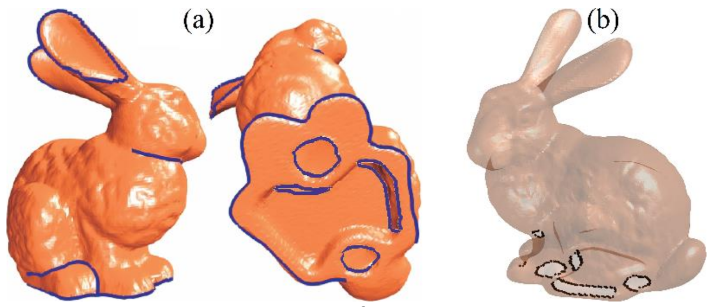

Figure 8 – Detected points on the Stanford bunny [29], with a method based on principal component analysis [11] (a) and with the BPD algorithm (b).

The algorithms were tested, using a computing machine based on an Intel® Core(TM) i7-6820HQ CPU (2.70GHz), with 32Gb of RAM. The tests were carried out through MATLAB® 2016a, running on the Windows 10 64-bit operating system. Table 1 reports quantitative outcomes obtained from the application of the BPD algorithm to the point clouds given in Figure 7. The initial rows of the table report the minimum, the mean, the maximum cloud resolution (β) and its standard deviation (σ), followed by the number of points in the clouds. Therefore the table gives the number of boundary points detected by the detection algorithm and the time taken (in milliseconds [ms]) for its complete execution. The elapsed time is always lower than 3 seconds for all examined point clouds.

Table 1 – Performances obtained from the application of the BPD algorithm.

Cloud	**a**	**b**	**c**	**d**	**e**
Min β [mm]	3.82	7.77	15.53	32.29	1.98
Mean β [mm]	4.19	8.44	16.87	34.52	3.71
Max β [mm]	4.85	9.36	18.98	40.42	34.86
SD (σ) [mm]	0.11	0.20	0.52	1.14	4.30
Number of points	12309	3095	791	203	7458
Boundary points	632	315	156	72	397
Detection time [ms]	2606	944	340	289	1892
Ω [% of β_i]	25%	24%	27%	25%	26%
Ψ [% of β_i]	80%	113%	84%	157%	75%

The point clouds obtained through some 3D scanning methods are affected by out-of-plane noise, meaning that the sampled points deviate from their ideal version lying on the surface. It is important to estimate how much the proposed detection algorithm is tolerant to such noise. Therefore the detection algorithm was repeatedly applied to versions of the point clouds with increasing random noise added to the original points. The noisy clouds were artificially obtained by moving each point along the normal direction of the local k-neighbourhood best fit plane. Each point was moved by distances equal to a percentage of the local cloud resolution. Starting from 0%, the noise percentage was increased by 1% at every repetition of the detection algorithm. The maximum percentage value after which at least one of the boundary points is not detected (false negative) is denoted as Ω. The value of Ω is around 25% for all the clouds of Figure 7. If the noise percentage is increased above Ω, some internal points of the cloud may be labelled as boundary points (false positives). The maximum percentage value after which at least one of the internal points is labelled as boundary point is denoted as Ψ. The value of Ψ in Table 1 are all above 75%. Therefore, the detection algorithm is very robust for out-of-plane noise lower than 25% of the cloud resolution and it can produce satisfactory results when the noise is lower than circa

75%. With noise values between 25% and 75% of the cloud resolution, the detection algorithm will miss some boundary points but no outliers will be generated.

Figure 9 compares, for the 16mm average resolution point cloud, polynomial fitting and B-spline edges of 2nd and 3rd order with the edges reconstructed through the new RBS approach (as described within the thickest dashed perimeter in Figure 6b). It is evident that the polynomial fitting produces unsatisfactory edges, especially for the internal boundaries. B-spline edges present excessive wrinkling, thus poor smoothing.

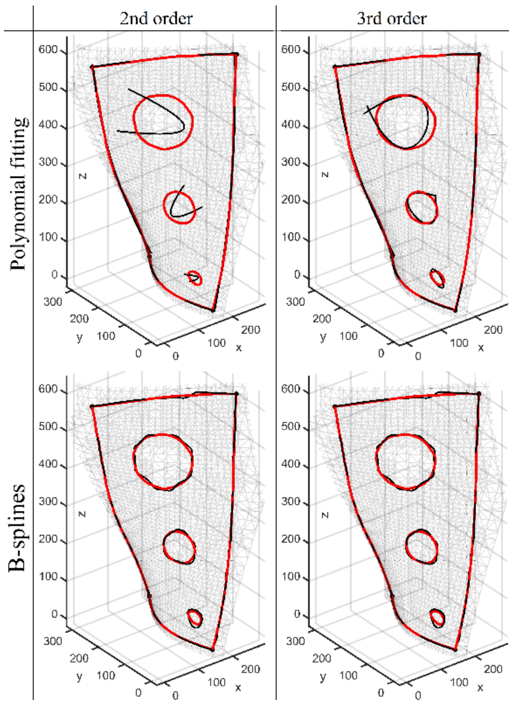

Figure 9 – Comparison of reconstructed edges (red line) with polynomial fitting and B-spline edges (black line) of 2nd and 3rd order.

Table 2 gives quantitative results on the performance of the RBS algorithm, compared to 3rd order polynomial fitting and B-splines, for all point clouds in Figure 7.

Table 2 – Performances of the RBS algorithm, compared to 3rd order polynomial fitting.

	Cloud	a	b	c	d	e
New method	**Reconstruction time [ms]**	528	305	143	71	385
	Mean dist. to points [mm]	0.41	0.84	0.96	1.38	0.22
	Max dist. to points [mm]	2.17	4.16	6.73	3.76	2.43
	Mean dist. to mesh [mm]	0.07	0.25	0.95	1.20	0.73
	Max dist. to mesh [mm]	0.80	1.68	4. 51	4.23	5.77
Pol. fitting	**Reconstruction time [ms]**	20	19	17	15	19
	Mean dist. to points [mm]	3.61	3.78	4.95	5.64	3.83
	Max dist. to points [mm]	34.8	33.0	32.7	40.0	30.1
	Mean dist. to mesh [mm]	1.00	0.89	1.08	1.43	1.27
	Max dist. to mesh [mm]	7.03	5.98	6.56	7.70	12.5
B-Splines	**Reconstruction time [ms]**	102	60	39	25	69
	Mean dist. to points [mm]	≈0	≈0	≈0	≈0	≈0
	Max dist. to points [mm]	≈0	≈0	≈0	≈0	≈0
	Mean dist. to mesh [mm]	0.05	0.11	0.24	0.46	0.45
	Max dist. to mesh [mm]	0.78	1.12	1.35	3.08	3.30

Although the execution time of the new FFT-based algorithm is always one order of magnitude higher than the time for polynomial fitting and for B-spline computation, the smooth edges produced by the new approach fit the surface contour better than polynomial fitting and B-splines. The table reports the mean, maximum and standard deviation (STD) values for the distances between the boundary points and the smoothed edges and for the distance between the reconstructed surface mesh and the reconstructed edges. In average, the reconstructed edges computed through the new approach fit the boundary points 4.7 times better than the 3rd order polynomial edges. Moreover they follow the reconstructed surface mesh contour 77% better than the polynomial fitting edges. Although the B-splines seem to follow the reconstructed mesh better than the new approach, 3rd order B-splines do not produce significant smoothing of the original boundary points; the distance between the B-splines and the points being circa equal to 0 is not a sign of good performance. The accuracy with which the polynomial and B-spline edges fit the boundary points and the reconstructed surface can be improved by increasing the order of the functions, but this impacts on the time required to compute the function coefficients. Moreover high order polynomials often suffer from severe ringing between the data points. The RBS approach reconstructs the optimum edges without the need to specify any parameter, unlike the function order in the polynomial and the B-spline fitting.

5. Conclusions

Tessellated surfaces generated from point clouds typically show inaccurate and jagged boundaries. This can lead to tolerance errors and problems such as machine judder if the model is used for ongoing manufacturing applications. This is the reason why many existing commercial computer-aided manufacturing (CAM) applications are not able to use tessellated models. This work presented novel algorithms to refine the boundary of meshed surfaces obtained from 3D scanning point cloud data. The BPD algorithm allows the unlabelled point of a surface point cloud to be grouped into two groups: boundary points and internal points. Existing detection techniques are optimized to detect points belonging to sharp edges and creases. The BPD algorithm is targeted to the detection of boundary points and it is able to do this better than the existing methods. The RBS algorithm identifies the boundary corners and divides each closed boundary into the constituting edges. Every edge is smoothed through spatial FFT-based filtering. A crucial advantage of the introduced algorithms is that they are not based on any threshold values that can be suitable for some point cloud but not suitable for others. The FFT-based edge reconstruction eliminates the problem of defining a specific polynomial function order for optimum polynomial curve fitting. The algorithms were tested to analyse the results and measure the execution time for point clouds generated from laser scanned measurements on a turbofan engine turbine blade with varying numbers of member points. Through adding artificial noise it has been demonstrated that the BPD algorithm is very robust for out-of-plane noise lower than 25% of the cloud resolution and it can produce satisfactory results when the noise is lower than circa 75%. With noise values between 25% and 75% of the cloud resolution, the detection algorithm will miss some boundary points but no outliers will be generated. Quantitative results on the performance of the RBS algorithm were also presented. The reconstructed edges computed through the new approach fit the boundary points by a factor of 4.7 times better than polynomial edges. Moreover they follow the reconstructed surface mesh contour with an improvement of 77% compared to the polynomial fitting edges.

Acknowledgements

This work is part of the Autonomous Inspection in Manufacturing and Re-Manufacturing (AIMaReM) project, funded by the UK Engineering and Physical Science Research Council (EPSRC) through the grant EP/N018427/1. The authors also wish to thank Dr Maxim Morozov for acquiring the point cloud of the turbofan engine turbine blade considered in this paper.

References

[1] B. Curless, "From range scans to 3D models," *ACM SIGGRAPH Computer Graphics,* vol. 33, pp. 38-41, 1999.
[2] M. Levoy and T. Whitted, *The use of points as a display primitive*: University of North Carolina, Department of Computer Science, 1985.
[3] T. Beard, "Machining From STL Files," *Modern Machine Shop,* vol. 69, pp. 90-99, 1997.
[4] C. Mineo, S. G. Pierce, P. I. Nicholson, and I. Cooper, "Introducing a novel mesh following technique for approximation-free robotic tool path trajectories," *Journal of Computational Design and Engineering*, 2017.
[5] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine*, et al.*, "A survey of surface reconstruction from point clouds," in *Computer Graphics Forum*, 2017, pp. 301-329.
[6] T. Hinks, H. Carr, L. Truong-Hong, and D. F. Laefer, "Point cloud data conversion into solid models via point-based voxelization," *Journal of Surveying Engineering,* vol. 139, pp. 72-83, 2012.
[7] L. Truong-Hong, D. F. Laefer, T. Hinks, and H. Carr, "Flying voxel method with Delaunay triangulation criterion for façade/feature detection for

computation," *Journal of Computing in Civil Engineering*, vol. 26, pp. 691-707, 2011.

[8] B. Delaunay, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, pp. 1-2, 1934.

[9] F. Calakli and G. Taubin, "SSD: Smooth signed distance surface reconstruction," in *Computer Graphics Forum*, 2011, pp. 1993-2002.

[10] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 32, p. 29, 2013.

[11] S. Gumhold, X. Wang, and R. S. MacLeod, "Feature Extraction From Point Clouds," in *IMR*, 2001.

[12] C. Weber, S. Hahmann, and H. Hagen, "Methods for feature detection in point clouds," in *OASIcs-OpenAccess Series in Informatics*, 2011.

[13] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," in *ACM transactions on graphics (TOG)*, 2005, pp. 544-552.

[14] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression," in *Computer Graphics Forum*, 2009, pp. 493-501.

[15] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, pp. 469-483, 1996.

[16] N. Akkiraju, H. Edelsbrunner, M. Facello, P. Fu, E. Mücke, and C. Varela, "Alpha shapes: definition and software," in *Proceedings of the 1st International Computational Geometry Software Workshop*, 1995, p. 66.

[17] L. Kettner, S. Näher, J. E. Goodman, and J. O'Rourke, "Two computational geometry libraries: LEDA and CGAL," in *Handbook of Discrete and Computational Geometry*, ed: Chapman & Hall/CRC, 2004, pp. 1435-1463.

[18] D. Bazazian, J. R. Casas, and J. Ruiz-Hidalgo, "Fast and robust edge extraction in unorganized point clouds," in *Digital Image Computing: Techniques and Applications (DICTA), 2015 International Conference on*, 2015, pp. 1-8.

[19] M. Weinmann, B. Jutzi, S. Hinz, and C. Mallet, "Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 286-304, 2015.

[20] N. L. Biggs, "The roots of combinatorics," *Historia Mathematica*, vol. 6, pp. 109-136, 1979.

[21] A. Schrijver, "On the history of combinatorial optimization (till 1960)," *Handbooks in operations research and management science*, vol. 12, pp. 1-68, 2005.

[22] H. Ni, X. Lin, X. Ning, and J. Zhang, "Edge detection and feature line tracing in 3d-point clouds by analyzing geometric properties of neighborhoods," *Remote Sensing*, vol. 8, p. 710, 2016.

[23] S. Arlinghaus, *Practical handbook of curve fitting*: CRC press, 1994.

[24] D. Johnson and R. P. D. Williams, "Methods of Experimental Physics: Spectroscopy," ed: Academic Press, New York, 1976.

[25] G. D. Knott, *Interpolating cubic splines* vol. 18: Springer Science & Business Media, 2012.

[26] C. de Boor, "On the convergence of odd-degree spline interpolation," *Journal of approximation theory,* vol. 1, pp. 452-463, 1968.
[27] R. C. Gonzales and R. E. Woods, "Digital Image Processing, Addison & Wesley Publishing Company," *Reading, MA,* 1992.
[28] J.-P. Monchalin, C. Neron, P. Bouchard, and R. Heon, "Laser-ultrasonics for inspection and characterization of aeronautic materials," *Journal of Nondestructive Testing & Ultrasonics(Germany),* vol. 3, p. 002, 1998.
[29] G. Turk and M. Levoy, "The Stanford Bunny, the Stanford 3D Scanning Repository," ed, 1994.
[30] G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994, pp. 311-318.