Journal of Computational Design and Engineering 4 (2017) 192–202

Contents lists available at ScienceDirect

Journal of Computational Design and Engineering

journal homepage: www.elsevier.com/locate/jcde

Introducing a novel mesh following technique for approximation-free robotic tool path trajectories ☆

Carmelo Mineo [a,*], Stephen Gareth Pierce [a], Pascual Ian Nicholson [b], Ian Cooper [b]

[a] Department of Electronic and Electrical Engineering, University of Strathclyde, Royal College Building, 204 George Street, Glasgow G1 1XW, UK
[b] TWI Technology Centre (Wales), Harbourside Business Park, Harbourside Road, Port Talbot SA13 1SB, UK

ARTICLE INFO

Article history:
Received 7 November 2016
Received in revised form 18 January 2017
Accepted 25 January 2017
Available online 16 February 2017

Keywords:
Tool path generation
Mesh following technique
Triangular meshes
Robotics
NDT

ABSTRACT

Modern tools for designing and manufacturing of large components with complex geometries allow more flexible production with reduced cycle times. This is achieved through a combination of traditional subtractive approaches and new additive manufacturing processes. The problem of generating optimum tool-paths to perform specific actions (e.g. part manufacturing or inspection) on curved surface samples, through numerical control machinery or robotic manipulators, will be increasingly encountered. Part variability often precludes using original design CAD data directly for toolpath generation (especially for composite materials), instead surface mapping software is often used to generate tessellated models. However, such models differ from precise analytical models and are often not suitable to be used in current commercially available path-planning software, since they require formats where the geometrical entities are mathematically represented thus introducing approximation errors which propagate into the generated toolpath. This work adopts a fundamentally different approach to such surface mapping and presents a novel Mesh Following Technique (MFT) for the generation of tool-paths directly from tessellated models. The technique does not introduce any approximation and allows smoother and more accurate surface following tool-paths to be generated. The background mathematics to the new MFT algorithm are introduced and the algorithm is validated by testing through an application example. Comparative metrology experiments were undertaken to assess the tracking performance of the MFT algorithms, compared to tool-paths generated through commercial software. It is shown that the MFT tool-paths produced 40% smaller errors and up to 66% lower dispersion around the mean values.

1. Introduction

Modern Computer-Aided Design (CAD) is used extensively in composite manufacture. Where it was once necessary to construct large items from many smaller parts, Computer-Aided Manufacturing (CAM) now allows these large items to be produced easily from one piece of raw material (through traditional subtractive approaches, or built up using more recent additive manufacturing processes (Gibson, Rosen, & Stucker, 2010). As a result, large components with complex geometries are becoming very common in modern structures.

Production engineers often face the problem of generating optimum tool-paths to perform specific actions on curved surfaces through numerical control machinery or robotic manipulators. The requirements of surface spray painting, surface coating, or Non-Destructive Testing (NDT) can be quite challenging to meet for complex shapes, especially when 100% coverage of the surfaces is required (Andulkar & Chiddarwar, 2015; Martin, 1967).

When working directly from available CAD models, there are a large number of Off-Line Programming (OLP) software packages available commercially that satisfy the requirements for such tool-path generation (e.g. MasterCAM®, Delcam®, Delmia®). However, when the original CAD model of the components is not available, photogrammetry or laser scanning must be used to create a point cloud of the surfaces of interest (Chikofsky & Cross, 1990; Varady, Martin, & Cox, 1997). There are also situations with composites

Abbreviations: CAD, Computer Aided Design; CAM, Computer Aided Manufacturing; NDT, Non-Destructive Testing; OLP, Off-Line Programming; STL, Standard Tessellation Language; MFT, Mesh Following Technique; CNC, Computer Numerical Control; NURBS, Non-Uniform Rational Basis Spline; CMM, Coordinate Measuring Machines; GUI, Graphical User Interface; SD, Standard Deviation; TOF, Time-Of-Flight.

☆ Peer review under responsibility of Society for Computational Design and Engineering.

* Corresponding author.
E-mail address: carmelo.mineo@strath.ac.uk (C. Mineo).

http://dx.doi.org/10.1016/j.jcde.2017.01.002
2288-4300/

manufacturing, compared to conventional light alloy materials, where the use of surface mapping metrology is required, even when the original CAD model is available. This situation can arise due to the inherent process variability associated with composites manufacture. Parts that are designed as identical may be affected by distortions when removed from the mould and may exhibit significant deviations from CAD (Zahlan & O'neill, 1989). These effects present a significant challenge for the execution of successive production operations.

When considering such parts, surface mapping leads, through the collection of point cloud data, to the generation of meshed CAD models of the parts (Fabio, 2003). Such models differ from precise analytical models, where all geometrical entities and spatial relationships are described analytically. Meshed CAD models are usually saved as Standard Tessellation Language (STL) files. The STL format is widely used for rapid prototyping and computer-aided manufacturing (Szilvśi-Nagy & Matyasi, 2003). The format only describes the surface geometry of a three-dimensional object without any representation of colour, texture or other common CAD model attributes. Whilst the conversion of analytical geometries into meshed surfaces is straightforward, the reverse process of conversion of a STL file into an analytical CAD model is challenging and time-consuming (Fabio, 2003).

Meshes represent 3D surfaces as a series of discreet facets, much as pixels represent an image with a series of coloured points. If the facets or pixels are small enough, the image appears smooth. Yet, if the surface is zoomed in enough, it is possible to see the pixelization or granularity and that the object is not locally smooth and continuous. This can lead to problems in the robot path creation (e.g. discontinuities and gaps) and explains the reason why existing commercial path-planning software requires precise part models, where the surfaces are mathematically represented.

This paper presents a novel algorithm based on a Mesh Following Technique (MFT) for the generation of tool-paths from STL models, suitable to overcome the difficulties encountered with current software applications that rarely support tessellated models as the input format for their embedded path-planning options.

2. Standard analytical approach

There is a wide range of algorithms suitable to generate boundary-conformed tool-paths for curved surfaces. Several studies have produced tool path generation methods for Computer Numerical Control (CNC) machining from STL models (Choi, Lee, Hwang, & Jun, 1988; Hwang, 1992; Jun, Kim, & Park, 2002; Ren, Yau, & Lee, 2004). However, these methods generate tool paths by approximating the polyhedral models and most of them focus on tool-paths for CNC machinery with limited number of axes (e.g. three-axis). A typical standard approach (Wang, Zhang, Scott, & Hughes, 2011) to convert tessellated STL surfaces into analytic surfaces is to approximate using Non-Uniform Rational Basis Spline (NURBS) or polynomial reconstruction. NURBS surfaces are mathematical representations of curves and surfaces; they are capable of representing complex free form surfaces that are inherently smooth. NURBS can be easily converted to meshes at any time, in the same way that one can easily take a digital image of an object with a camera. Conversely, going from meshes to NURBS is like trying to reconstruct the object from a pixelated digital image – it is a much more difficult task. NURBS surfaces are generated by a series of NURBS curves in two directions (called U and V) interpolated to create a surface. There are no quick automatic methods to convert tessellated surfaces to NURBS. Some CAD applications (e.g. Rhinoceros® by McNeel) include conversion tools, but consider only the simplest case of NURBS surfaces – the so called *bilinear surfaces* defined by 1 degree NURBS curves (i.e. lines) in both directions (Piegl & Tiller, 2012).

Curve fitting is the process of approximating a pattern of points with a mathematical function (Arlinghaus, 1994). Fitted curves can be used to infer values of a function where no data are available (Johnson & Williams, 1976), overcoming the discretization of pixelated or tessellated models. Regression analysis provides robust statistical tools to estimate how much uncertainty is present in a curve that is fit to discreet data points (Freund, Wilson, & Sa, 2006). The goal of regression analysis 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 a nth degree polynomial function, yielding the general polynomial regression model based on the truncated Taylor's series:

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n + \varepsilon \tag{1}$$

where ε is a random error with mean zero. Conveniently, these models are all linear from the point of view of estimation, since the regression function is linear in terms of the unknown coefficients $a_0, a_1, \ldots, a_n$. Therefore, for least squares analysis, the computational and inferential problems of polynomial regression can be completely addressed using the techniques of multiple regression. This is done by treating $x, x^2, \ldots, x^n$, as being distinct, independent variables in a multiple regression model.

An initial attempt was made to approximate meshes with polynomial analytical surfaces. A MATLAB® surface fitting toolbox (D'Errico, 2010) was used. Given the vertices of the tessellated model and the order of the target polynomial function, *the fitting algorithm* provides the coefficients of the fitting function. The order of the polynomial function can be progressively increased until the approximation error falls below a set threshold, at the expense of increasing the computation time. At each iteration, the residual errors and the R-squared parameter of the regression can be computed. The R-squared parameter is a statistical measure of how close the data are to the fitted regression surface. It is also known as the coefficient of determination. R-squared is always between 0% and 100%, and indicates how well the polynomial model explains the variability of the data around its mean. However, it is possible to obtain high R-squared values also for a model that does not fit the data well. Therefore, the analysis of the maximum of the residual errors is more useful in practice. Monitoring the residual errors is also useful to assure the stability of the approximation algorithm; the increase of the polynomial order can be ceased when the residual errors begin to diverge.

The use of this approach highlighted some important limitations. Even classic primitive geometric surfaces can be surprisingly difficult to approximate, leading to coarse approximation errors. The iterative polynomial approximation was applied to the tessellated surface of one quarter of an ellipsoid with semi-major axis of 1 m and semi-minor axes of 0.5 m (Fig. 1a). The surface is represented by a triangular mesh with 1914 vertices and 3557 triangles. The maximum approximation error decreases up to the 30th order function, reaching a minimum value of 8.9 mm, before starting diverging to higher values. Fig. 1b and c shows respectively the 3rd order and the 30th order fitting surfaces, superposed to the mesh vertices.

Fig. 2 shows the maximum error, the R-squared parameter and the computation time, plotted against the order of the fitting polynomial function. All the algorithms presented in this paper were implemented in MATLAB® codes and tested using a Windows 10 based computer with 2.7 GHz Intel i7 processor. The computation time increased exponentially with the polynomial order, jumping from few milliseconds for the approximations with the lowest

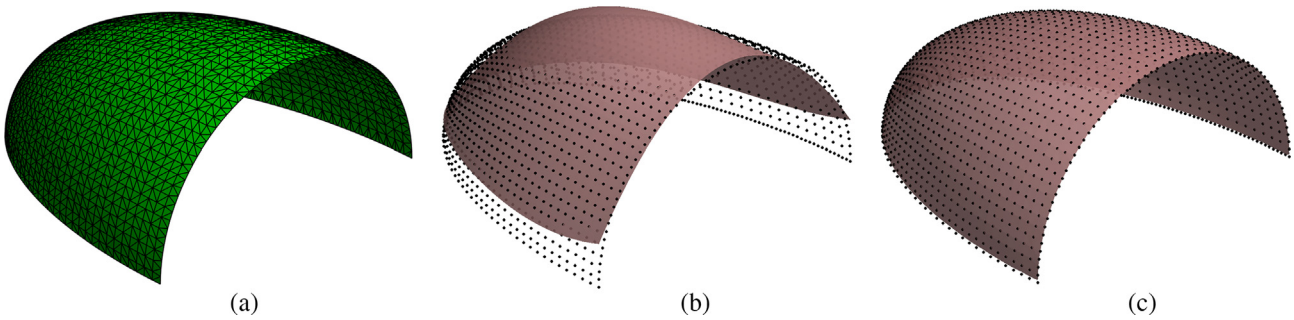


Fig. 1. Tessellated surface of one quarter of ellipsoid (a), approximation with 3rd order polynomial function (b) and with 30th order function (c).

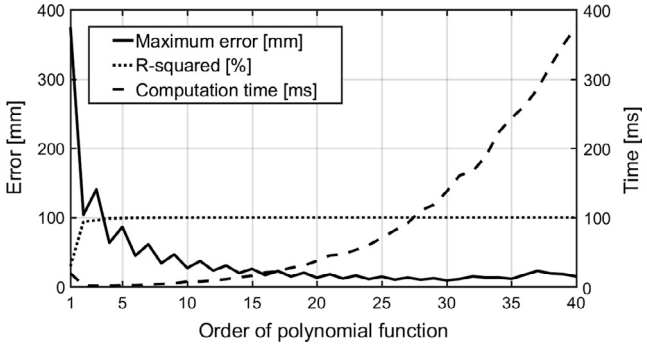


Fig. 2. Maximum error, *R*-squared parameter and computation time, plotted against the order of the fitting polynomial function.

orders to almost 4 s for the computation of a 40th order approximation.

Moreover, the approximation of a meshed surface with a polynomial surface is only possible when the surface can be mathematically described by a surjective function (Hassett & Tschinkel, 2006). A function, $z = f(x, y)$ with X–Y domain and codomain in Z, is defined surjective (or a surjection) if every element z in Z has a corresponding x–y couple such that $z = f(x, y)$. The function f may map more than one pair of independent variables (x, y) to the same element of Z, but not the opposite. Fig. 3 shows an example of surjective surface and non-surjective surface. The inverse of a surjective function is a not surjective function. As a result, the approximation of a meshed surface fails if the surface is not surjective and the approximation error is influenced by the orientation of the surface in the 3D Cartesian space.

Given these limitations, this work does not present any further investigation on polynomial reconstruction methods, since they only work for particular circumstances. Another path-planning approach for polyhedral machining was developed in 2005 (Yuwen, Dongming, & Haixia, 2006). This approach uses a harmonic map to parameterize the triangular meshes. The harmonic map based parameterization defines a map between regions on the 2D plane and the surface embedded in the 3D space, and enables this operation to be performed as easily as if the surface were flat. Unfortunately, likewise the polynomial regression, the method only works for surjective triangular meshes. Moreover, the interval between two subsequent passes of a cutter on the machined surface is approximated and can differ from the target value.

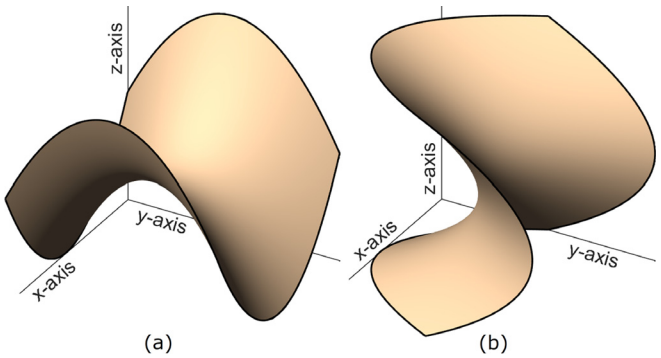


Fig. 3. Example of surjective surface (a) and non-surjective surface (b).

3. The mesh-following technique

A new approach, herein referred to as the Mesh Following Technique (MFT), is hereby introduced. The MFT overcomes the limitations of existing methods for the generation of tool-paths from meshed surfaces and it can be applied to CNC tool-paths, as well as robot path-planning and trajectory generation for Coordinate Measuring Machines (CMM). The approach is based on the idea that the triangular mesh of a given surface can be used as guide for path-planning algorithms to operate directly on the curved contour of the surface and remove the need for approximation using curve fitting. MFT can be used to find the coordinates of points on the mesh at specific distances from the edges and generate single curve trajectories as well as rasters or more sophisticated paths (all lying on the meshed surface).

The fundamentals of MFT are presented below. A 1.6 m^2 tessellated surface with three irregular holes with geometry mirroring a real winglet component sample is used as reference mesh (Fig. 4), to facilitate the comprehension of the MFT path-planning algorithms. The mesh consists of 4858 vertices, linked by 9189 triangles.

The explanation of the MFT fundamentals can start from a practical example: the generation of a curve parallel to one of the mesh external edges, at distance d from the edge. The vertical edge on the left hand side of the mesh (edge #1, as indicated in Fig. 4) is used for the following descriptions. The target distance between the reference edge and the new curve is set to $d = 100$ mm.

Since the reference edge is part of the boundary of a tessellated surface, it consists of segments, whose extremities coincide with pairs of vertices of the adjacent triangles. The directional vector of the ith segment (s_i) is given by $\vec{v}_i = [u_i, v_i, w_i]$, where u_i, v_i and w_i are the vector components along x, y, and z-axis. The middle point of the segment $\vec{v}_i$ is $P_i = (x_i, y_i, z_i)$. Therefore the plane π_i, perpendicular to the segment s_i for the point P_i, is mathematically represented by the equation:

$$\pi_i : (u_i \cdot x) + (v_i \cdot y) + (w_i \cdot z) + d_i = 0 \tag{2}$$

where d_i is equal to:

$$d_i = -(u_i \cdot x_i + v_i \cdot y_i + w_i \cdot z_i) \tag{3}$$

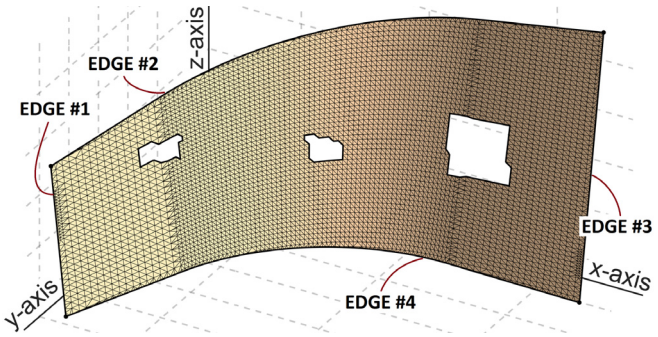


Fig. 4. Tessellated surfaces used for the description of the MFT algorithms.

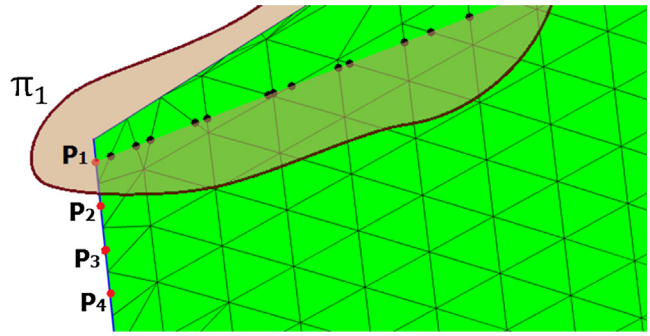


Fig. 5. Intersection points between mesh and plane for P_1.

For each segment of the reference edge it is possible to find the intersection points between the calculated plane and the edges of the triangles of the surface mesh. Fig. 5 shows the intersection points, originating from the plane relative to the first segment (s_1) of the reference edge and the edges of the tessellated mesh.

Starting from the ith point (P_i) and following the succession of the intersection points found on the relative ith plane (π_i), the curvilinear distance from the reference edge is calculated, though cumulating the distances between consecutive points. The curvilinear distance is the proper distance along the tessellated surface contour; its value is monitored and the addition of distances proceeds until the cumulative distance exceeded the set target d. The remaining intersection points are ignored. Since the last considered point is farther than the set distance and the second-last point is closer, a point that is exactly at the set distance is calculated through interpolation between the two points. This point lies on one of the mesh triangles (Fig. 6a). The process is repeated for all the segments in the reference edge (Fig. 6b). No point is found if all point distances are cumulated and the value of curvilinear distance remains smaller than d. The found points constitute a curve, parallel to the selected reference edge (Fig. 6c). Since all the points of the curve lie on the mesh triangles, the maximum deviation between the curve and the contour of the surface (represented by the triangular mesh) is equal to the deviation between the mesh and the sample surface. The algorithm does not introduce any additional approximation error. If the tessellated mesh is exported from a precise analytical CAD model, it will be sufficient to specify the maximum acceptable deviation during the exportation process; the described path-planning algorithm will inherit the same level of accuracy. On the other hand, if the mesh originates from a process of reverse engineering, the accuracy of the path generated through the algorithm depends on the accuracy of the metrology instrumentation used to map the surface (assuming that no smoothing filters are applied to mitigate the noise associated to the raw point cloud).

The extremities of the generated curve do not intersect the surface outer boundary; however, it is possible to use a simple method to extend the curve extremity segments to the boundary of the surface. Therefore, two additional points (one for each extremity) can be added to the curve, to make sure it starts and ends at the surface boundary (Fig. 7a and b).

The method is based on the calculation of the minimum distance between two skew lines (r and s) (see Fig. 7c). Let us consider the two lines containing the extremity segment (r) of the parallel curve and the neighbour segment (s) of the boundary. Each line can be represented by the start point of the relative segment and by the direction vector that links the start point to the end point:

$$r : R = (x_R, y_R, z_R); \vec{v_r} = (u_r, v_r, w_r) \tag{4}$$

$$s : S = (x_S, y_S, z_S); \vec{v_s} = (u_s, v_s, w_s) \tag{5}$$

Let us call $\overline{PQ}$ the generic segment from line r to line s. The length of $\overline{PQ}$ is equal to the minimum distance between the skew lines if it forms right angles with each of the two lines. In other words, $\overline{PQ}$ has to be perpendicular to both lines. Introducing the parameters t and h, the general coordinates of point P and Q are:

$$P = \begin{pmatrix} x_R + t \cdot u_r \\ y_R + t \cdot v_r \\ z_R + t \cdot w_r \end{pmatrix} \tag{6}$$

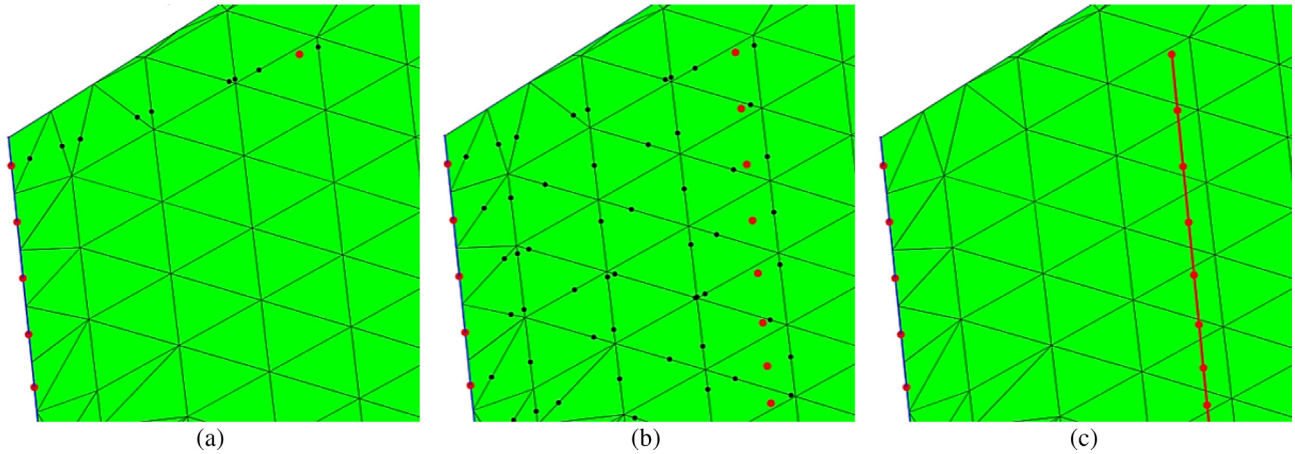


Fig. 6. Step-by-step process for the generation of a curve parallel to the reference edge.

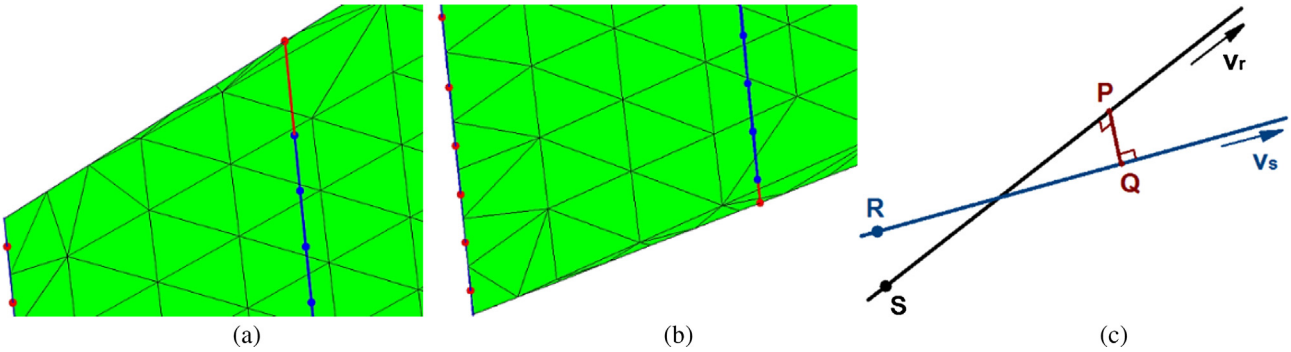


Fig. 7. Extension of trajectory to the surface boundary. First extremity (a) and second extremity (b). Minimum distance between two skew lines (c).

$$Q = \begin{pmatrix} x_S + h \cdot u_s \\ y_S + h \cdot v_s \\ z_S + h \cdot w_s \end{pmatrix} \tag{7}$$

Therefore, the segment is given by:

$$\overline{PQ} = \begin{pmatrix} x_S + h \cdot u_s - x_R - t \cdot u_r \\ y_S + h \cdot v_s - y_R - t \cdot v_r \\ z_S + h \cdot w_s - z_R - t \cdot w_r \end{pmatrix} \tag{8}$$

Applying the condition of perpendicularity between $\overline{PQ}$ and both lines:

$$\begin{cases} \overline{PQ} \cdot \vec{v_r} = 0 \\ \overline{PQ} \cdot \vec{v_s} = 0 \end{cases} \rightarrow \begin{cases} \begin{pmatrix} x_S + h \cdot u_s - x_R - t \cdot u_r \\ y_S + h \cdot v_s - y_R - t \cdot v_r \\ z_S + h \cdot w_s - z_R - t \cdot w_r \end{pmatrix} \cdot \begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} = 0 \\ \begin{pmatrix} x_S + h \cdot u_s - x_R - t \cdot u_r \\ y_S + h \cdot v_s - y_R - t \cdot v_r \\ z_S + h \cdot w_s - z_R - t \cdot w_r \end{pmatrix} \cdot \begin{pmatrix} u_s \\ v_s \\ w_s \end{pmatrix} = 0 \end{cases}$$

$$\rightarrow \begin{cases} (x_S + h \cdot u_s - x_R - t \cdot u_r) \cdot u_r + (y_S + h \cdot v_s - y_R - t \cdot v_r) \cdot v_r \\ \qquad + (z_S + h \cdot w_s - z_R - t \cdot w_r) \cdot w_r = 0 \\ (x_S + h \cdot u_s - x_R - t \cdot u_r) \cdot u_s + (y_S + h \cdot v_s - y_R - t \cdot v_r) \cdot v_s \\ \qquad + (z_S + h \cdot w_s - z_R - t \cdot w_r) \cdot w_s = 0 \end{cases} \tag{9}$$

The solution of the system of equations gives the values of the parameters t and h. The substitution of h into Eq. (7) gives the coordinates of point Q. This point belongs to the boundary of the surface and can be used to extend the parallel curve.

The resulting extended curve crosses the meshed surface from side to side. It is possible to use a searching algorithm to find the triangles of the mesh traversed by the curve (Fig. 8a).

Some robotized applications on complex geometries, like automated spray painting tasks or NDT probe scanning (e.g. through ultrasonic or eddy-current probes (Halmshaw, Honeycombe, & Hancock, 1991), require tool-paths made of multiple passes to achieve 100% coverage of the surface of interest. In these situations, raster scan paths are often used. For the generation of such tool-paths through the proposed MFT method, it is sufficient to iterate the algorithm described so far to covering the full extension of the meshed surface. The set of traversed triangles, shown in Fig. 8a, divides the mesh in two regions: the region that has already been swept by the algorithm, between the reference edge and the generated curve, and the remaining part of the mesh. The former region is identified (see Fig. 8b) and excluded from the domain of interest for the iteration of the algorithm. Therefore, the first generated curve is used as new reference edge to compute another parallel curve. The parallel curves can be equally spaced (if the target distance d is maintained constant) or with variable gaps between them (if d is changed between the iterations). Fig. 8c shows the result of the completed iteration of the algorithm, to achieve the full coverage of the meshed surface through equally spaced parallel curves. The arrows shown indicate the generated normals to the surface and these are used for the end effector tool orientation with respect to the sample surface.

Machines and robot manipulators with more than three degrees of freedom (DoF) can reach any point in the Cartesian 3D space (within their working environment) and also control the orientation of the tool. The coordinates of the trajectory points

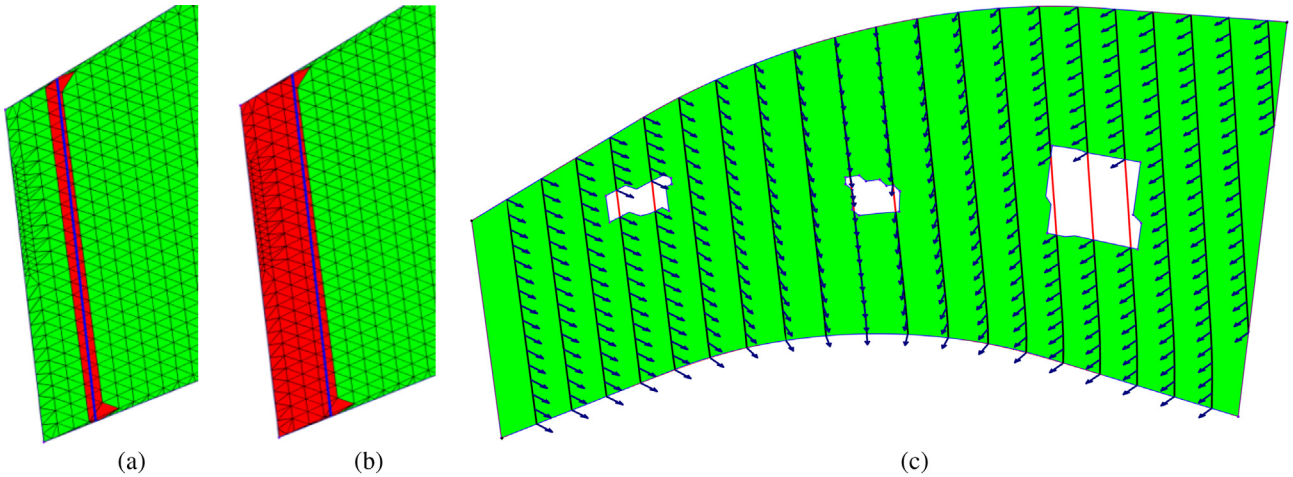


Fig. 8. Triangles crossed by the parallel curve (a), meshed region to be excluded (b) and completed iteration of algorithm (c), showing generated normals to the surface.

supply constrains for three DoF (x, y and z coordinates). In order to constrain the orientation of the tool relative to the surface, it is necessary to associate an orthogonal reference system to every point of the trajectory. The most practical axes to use, in order to construct such trajectory reference systems are the normal to the surface, the direction of travel and the mutually perpendicular vector tangential to the surface. Since every point of the obtained curves lies on the surface mesh, the perpendicular direction associated to each point is derived from the vectors normal to the mesh triangles. The cross product between the vector normal to the surface and the vector tangent to the trajectory generates the vector of the third axis of the orthogonal system. The constraint of the tool orientation, relative to the trajectory reference systems, leads to the definition of up to three other angular coordinates, expressed in the form of yaw, pitch and roll angles (Euler angles, typically denoted as α, β and γ).

Fig. 8c shows the normal vectors, associated to the points of the generated curves. The portions of the curves that cross the holes of the mesh are recognised. The intersection points between the curves and the inner boundaries of the surface are computed through the previously described method, based on the minimum distance between skew lines (see Eqs. (4)–(9)).

The MFT method is not limited to the generation of raster paths; it is suitable to be used in the development of very different path-planning strategies. The basic MFT capability of following the mesh contour to produce trajectories with the desired features, can be exploited in many ways. Fig. 9a shows the use of MFT to compute the position of a point (P) placed at distance of 800 mm from edge #3 and 200 mm from edge #4. This kind of single point computation can be valuable for several manufacturing operations on large complex geometries (e.g. drilling, tapping, spot welding, etc.). Fig. 9b shows the resulting trajectory, generated through a MFT function developed for generating spiral tool-paths; the barycentre (B) of the mesh central hole was used as origin of the spiral. The MFT approach maintained a spiral pitch of 100 mm throughout the whole mesh. In this case, the cutting planes, responsible to identify the fundamental intersection points lying on the edges of the triangles of the surface mesh, are planes passing through B and perpendicular to the diametric vectors of a circle centred in B.

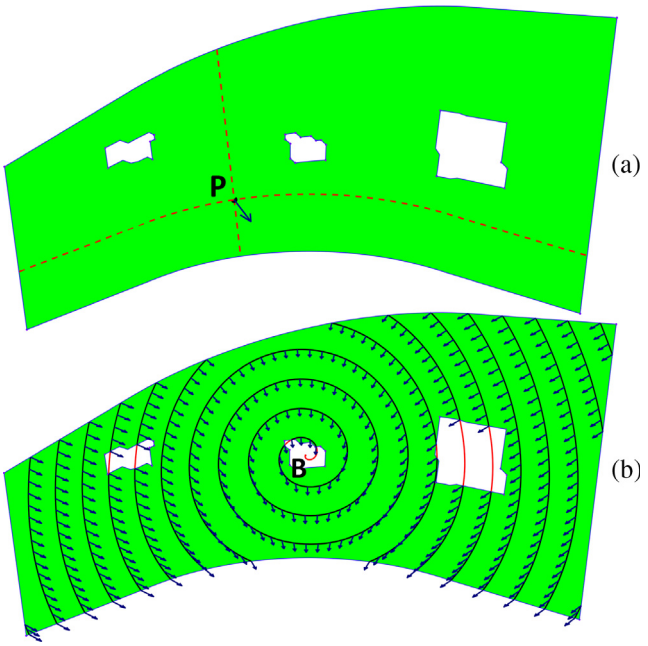


Fig. 9. Point at given distance from two edges (a) and spiral trajectory (b).

The implementation of the MFT method to generate the raster path, the single point and the spiral trajectory, produced execution times of 44.8 s, 11.0 s and 75.4 s, for the three tool-paths respectively.

4. Exploitation of the algorithm for flexible robot programming

This section of the paper presents an example of exploitation of the MFT algorithms, to compute suitable tool-paths for robotic non-destructive testing applications.

Six-axis robotic arms have traditionally been used in production lines to move the robot end-effector from one position to another for repetitive assembly and welding operations. In this scenario, where the exact trajectory between two points in the space is not too important, the teach pendant of a robot is used to manually move the end-effector to the desired position and orientation at each stage of the robot task. Relevant robot configurations are recorded by the robot controller and a robot programme is then written to command the robot to move through the recorded end-effector postures. More recently, accurate mechanical joints and control units have made industrial robotic arms flexible and precise enough for finishing tasks in manufacturing operations (Bogue, 2009). Robotic manipulators are highly complex systems and the trajectory accuracy of a machining tool has a huge impact on the quality and tolerances of the finished surfaces. As a result, many software environments have been developed by manufacturers, in order to help technicians and engineers to program complex robot tasks (Pan, Polden, Larkin, Van Duin, & Norrish, 2012). The use of such software platforms to program robot movements is known as Off-Line Programming (OLP). It is based on the 3D virtual representation of the complete robot work cell, the robot end-effector and the samples to be manipulated or machined. Conventional OLP is typically geared towards manufacturing applications where the task is the production of a specific component using conventional milling, drilling or trimming operations. For these manufacturing operations a precise CAD model is usually available.

Despite the large variety of path-planning options available, the existing commercial OLP software is not capable of working with tessellated surfaces. An example coming from the robotic deployed NDT of industrial components was selected to demonstrate the use of the new MFT method. The automated NDT inspection of complex geometries often faces the problem of generating inspection strategies for components whose CAD model is not available. Moreover, recent research has identified that robotic NDT requires a flexible and extensible robot programming approach that has the flexibility to allow future changes in the path planning to accommodate requirements of future NDT inspections (Mineo, Pierce, Nicholson, & Cooper, 2016). Although some limited applications for inspection delivery have been demonstrated (Haase, 2013), a series of serious inadequacies of commercial OLP applications motivate the use of the new MFT path-planning approach.

RoboNDT is a software platform tailored to the generation of tool-paths for the inspection of curved surfaces by 6-axis industrial robots (Mineo et al., 2016); it was developed by the same authors of this paper in 2015. RoboNDT is intended to be flexible and extendable to accommodate future system and robot developments. It is MATLAB® based, so it provided an easy route for the exploitation of the MFT algorithms.

Fig. 10a shows a picture of the robotic environment used to test the MFT tool-paths. The robotic cell contains two KUKA KR16 L6-2 robotic arms. Mounted on the robot end-effectors are water jet nozzles, which encapsulate ultrasonic phased array probes. Ultrasonic waves are much less attenuated when they propagate in water rather than air; therefore, the water column created by the nozzles was used to efficiently transmit the ultrasonic energy from

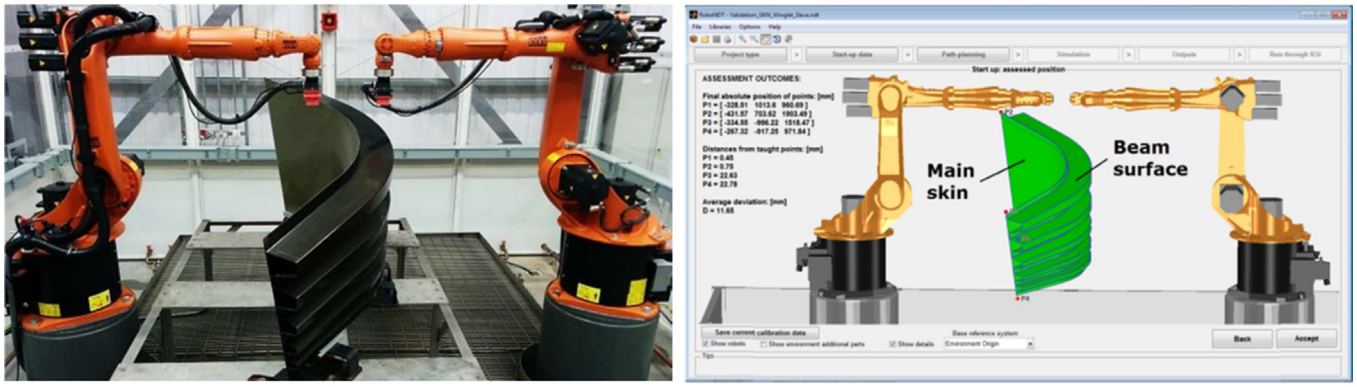


Fig. 10. Calculated position of the sample, waiting for the user to approve.

the probe to the specimen under inspection and receive the returning ultrasonic echoes. The specimen in Fig. 10 is a curved carbon fibre composite sample (a portion of an aerospace winglet). Fig. 10b shows the same robot environment as it was mapped out in RoboNDT, with the tessellated model of the sample correctly positioned through a sample position calibration procedure.

The MFT algorithms were embedded in RoboNDT, constituting the fundamental core of the RoboNDT tool-path generation capabilities. The potentialities of the MFT method were exploited through a structured Graphical User Interface (GUI) (Fig. 11). This enables the user to generate multiple inspection tool-paths (called *tasks*), of different types.

If the desired tool-path type is a raster, the parallel trajectories have to be linked to generate a single scanning raster path. The end of each raster pass is linked to the first point of the next pass, inserting a connecting path. At the end of this phase, the software adds the kinematic features to the tool-path. Acceleration and deceleration ramps characterise the robot end-effector speed pattern at the start and at the end point of each continuous portion of the tool-path. If α is the duration of the acceleration and deceleration ramps in a normalised time scale (t) and $v(t)$ is the normalised speed as a function of time, the following conditions are applied to obtain a continuous speed pattern:

$$\begin{cases} v(0) = 0 \\ v(\alpha) = 1 \\ v(1-\alpha) = 1 \\ v(1) = 0 \end{cases} \quad \begin{cases} v'(0) = 0 \\ v'(\alpha) = 0 \\ v'(1-\alpha) = 0 \\ v'(1) = 0 \end{cases} \tag{10}$$

The typical speed pattern is given in Fig. 12a. It is described by the function:

$$\begin{cases} v(t) = -\frac{2}{\alpha^3}t^3 + \frac{3}{\alpha^2}t^2 & \text{for } 0 \leqslant t \leqslant \alpha \\ v(t) = 1 & \text{for } a \leqslant t \leqslant (1-\alpha) \\ v(t) = \frac{2}{\alpha^3}(t+\alpha-1)^3 - \frac{3}{\alpha^2}(t+\alpha-1)^2 + 1 & \text{for } (1-\alpha) \leqslant t \leqslant 1 \end{cases} \tag{11}$$

When the tool-path continuous portion is not long enough to allow reaching of the regime speed, e.g. for short distances between two consecutive parallel curves, the following conditions replace the former ones:

$$\begin{cases} v(0) = 0 \\ v(0.5) = \beta \\ v(1) = 0 \end{cases} \quad \begin{cases} v'(0) = 0 \\ v'(0.5) = 0 \\ v'(1) = 0 \end{cases} \tag{12}$$

where β is a percentage of the target speed used for the raster scan. This parameter spans between 0 and 1 according to the length of the trajectory linking two consecutive lines of the scan path. Small values of β are used for short trajectories, to let the robot quietly abandon the end point of the finished line and reach the starting point of the next line. The speed function results:

$$v(t) = 16\beta t^4 - 32\beta t^3 + 16\beta t^2 \quad \text{for } 0 \leqslant t \leqslant 1 \tag{13}$$

The typical pattern of the speed is given in Fig. 12b.

Fig. 13 shows the inspection tool-paths generated through RoboNDT for the experimental tests. For the sake of testing the tool-paths with surfaces curving in different directions, the main skin of the winglet and the top surface of one of its back wall beams (see Fig. 10b) were considered for path-planning. The main skin surface had an area of 1.6 m^2; this surface was identical to the surface used above for the description of the MFT method (except for the absence of irregular holes). The beam surface had an area of 0.5 m^2. The inspection tool-paths were raster scans with a raster step of 29.4 mm, equal to the width of the ultrasonic phased array probe active area.

The output function of the software translated the generated tool-path into a set of command coordinates packets that were interpreted by the robot controllers. Each robot pose was represented by a vector, $p = [x, y, z, \alpha, \beta, \gamma]^T$, containing the three Cartesian coordinates of a given position and the roll (α), pitch (β) and yaw (γ) angles of the end-effector orientation for that position. The conversion of the normal vector components (N_X, N_Y, N_Z) into the angular coordinates was based on the following rotational matrix:

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} = -\begin{bmatrix} V_X & T_X & N_X \\ V_Y & T_Y & N_Y \\ V_Z & T_Z & N_Z \end{bmatrix} \tag{14}$$

The normal vector components populate the third column of the matrix. The second column contains the tangential vector, representing the direction of travel calculated as:

$$\mathbf{T} = \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix} = \begin{bmatrix} \frac{dx}{\sqrt{dx^2+dy^2+dz^2}} \\ \frac{dy}{\sqrt{dx^2+dy^2+dz^2}} \\ \frac{dz}{\sqrt{dx^2+dy^2+dz^2}} \end{bmatrix} \tag{15}$$

where dx, dy and dz are the gradients of the trajectory in the three dimensions. The first column contains the bi-normal vector:

$$\mathbf{V} = \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} = \begin{bmatrix} N_Y T_Z - N_Z T_Y \\ N_Z T_X - N_X T_Z \\ N_X T_Y - N_Y T_X \end{bmatrix} \tag{16}$$

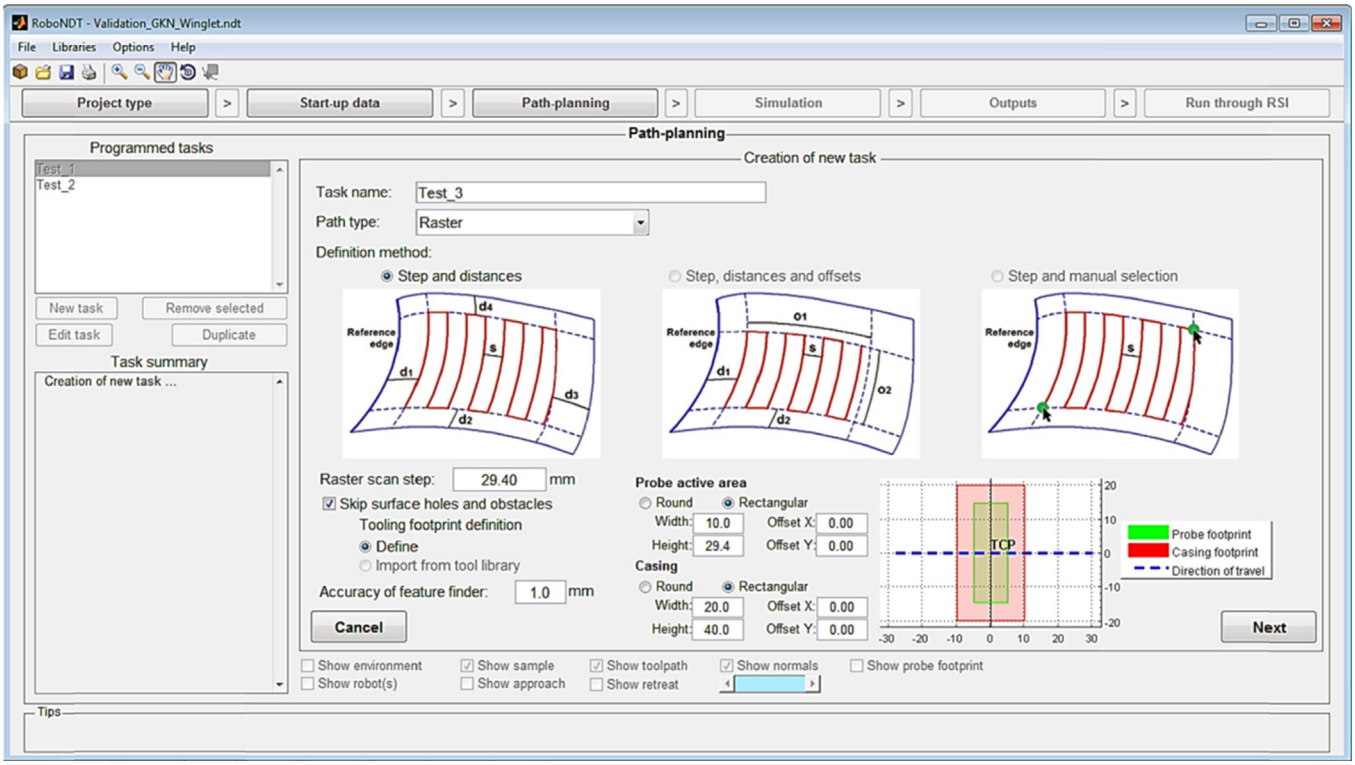


Fig. 11. RoboNDT path-planning GUI, developed to exploit the potentialities of the MFT method.

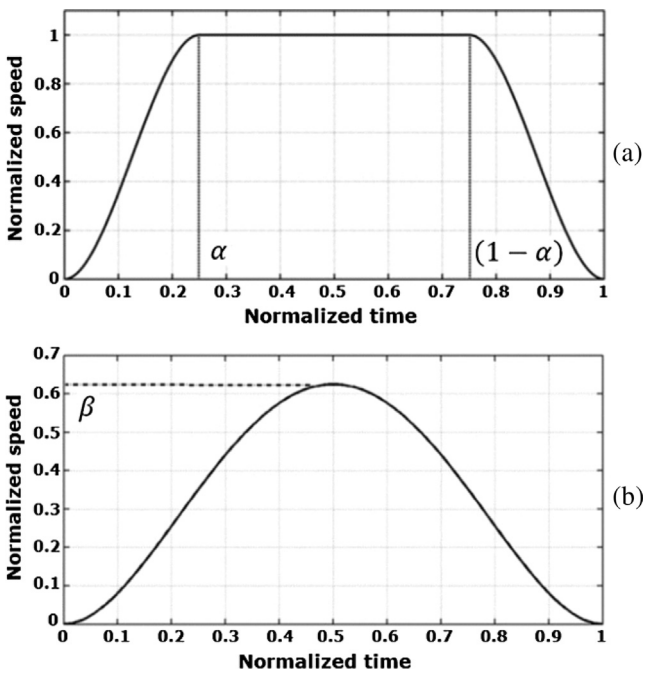


Fig. 12. Typical speed pattern for a long continuous curve (a) and a short one (b).

Thus the angular coordinates (α, β, γ) are calculated through the following formulation:

$$\beta = \operatorname{atan2}\left(\frac{-R_{31}}{\sqrt{R_{11}^2 + R_{21}^2}}\right) \tag{17}$$

$$\alpha = \begin{cases} 0, & \text{for } |\beta| = \pi/2 \\ \operatorname{atan2}\left(\frac{R_{21}}{R_{11}}\right), & \text{for } |\beta| \neq \pi/2 \end{cases} \tag{18}$$

$$\gamma = \begin{cases} \frac{B}{abs(B)} \cdot \operatorname{atan2}(R_{12}), & \text{for } |\beta| = \pi/2 \\ \operatorname{atan2}\left(\frac{R_{32}}{R_{33}}\right), & \text{for } |\beta| \neq \pi/2 \end{cases} \tag{19}$$

A six-axis robot can theoretically reach any point within its working envelope using 8 different configurations of the joints (Corke, 2011). The inverse kinematics of six-axis robots, based on a geometric approach (Weber, 2009), was used to allow controlling of the robot tool-path through the axis coordinates, rather than Cartesian coordinates. This provided a pathway to avoid singularity problems, by selecting the most suitable kinematic configuration to the execution of the whole robot task.

5. Accuracy results

The tool-paths were executed with robot end-effector velocities of 100 mm/s and 300 mm/s, maintaining the same acceleration limit of 500 mm/s^2. The robot positional feedback was compared to the commanded positions, to make sure the MFT tool-paths were output correctly and accurately followed by the robot hardware. The positional error was defined as the distance between the commanded tool centre points (TCPs) and the actual reached points as measured by the robot encoders. The orientation error was defined as the mismatch angle between the commanded rotation matrix and the rotation matrix computed from the feedback roll, pitch and yaw angles.

Table 1 shows the distribution maps of position and orientation errors for all MFT tool-paths generated through RoboNDT. For the sake of helping the comparison, the same colour scale is maintained where possible.

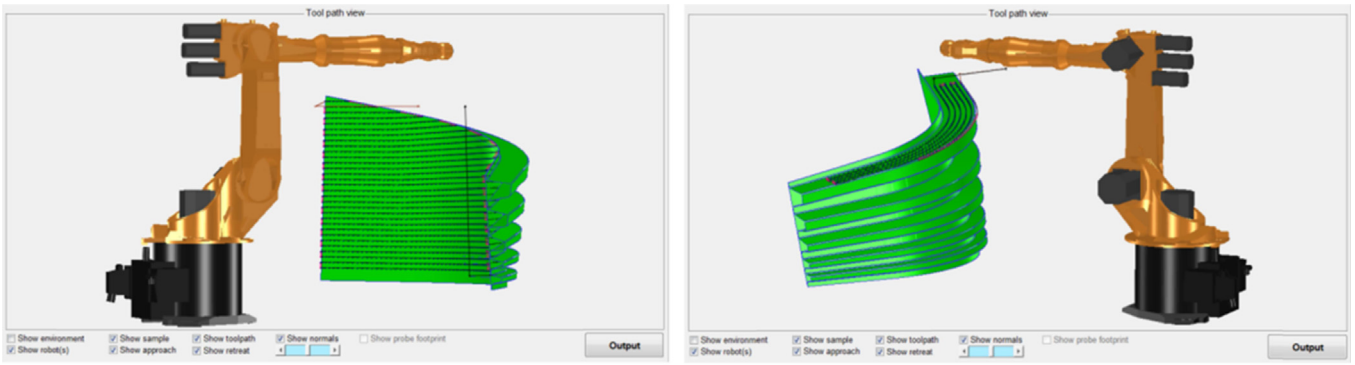


Fig. 13. Evaluation of generated tool-paths.

Table 1
Maps of position and orientation errors.

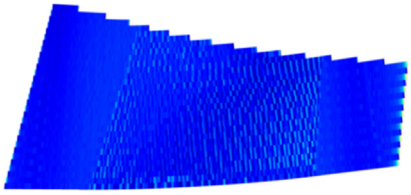
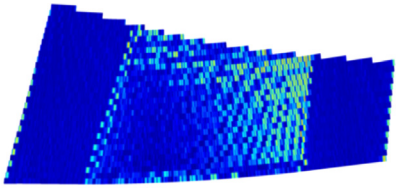
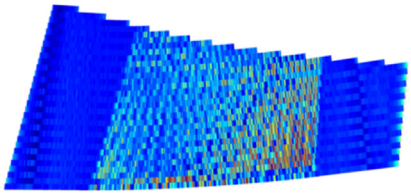
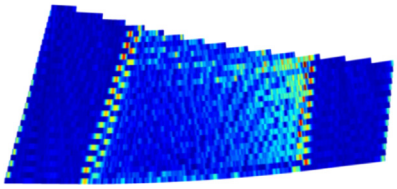
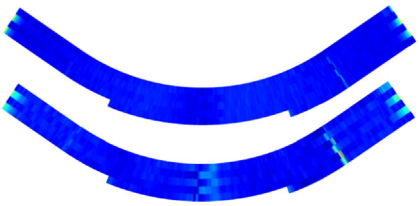
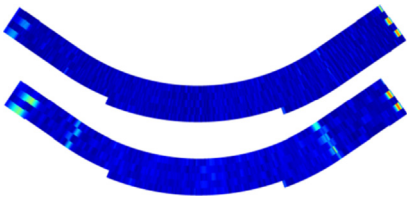
		Position error (mm)	Orientation error (degrees)
Main skin	100 mm/s		
	300 mm/s		
Beam	100 mm/s		
	300 mm/s		
		0 0.5 1 1.5 2	0 0.05 0.1 0.15 0.2 0.25

Table 2
Maximum, mean error and Standard Deviation (SD).

		Robot speed: 100 mm/s		Robot speed: 300 mm/s	
		Main skin	Beam surface	Main skin	Beam surface
Position error [mm]	Max	1.368	1.181	2.696	1.529
	Mean	0.279	0.259	0.427	0.297
	SD	0.106	0.086	0.291	0.142
Orientation error [degrees]	Max	0.205	0.257	0.292	0.235
	Mean	0.028	0.017	0.033	0.024
	SD	0.029	0.014	0.033	0.023

Table 2 reports the maximum and mean value and the Standard Deviation (SD) of the positional and rotational errors. The SD values are useful to quantify the amount of variation around the mean value. The position error remained below 2.7 mm and the orientation error below 0.29 degrees. As it is expected, faster speeds produce bigger errors because of the inertial effects affecting the robotic motion.

The variability of the standoff between the probe and the surfaces is shown in Table 3 with maps derived from the Time-Of-Flight (TOF) of the ultrasonic wave reflected from the scanned surface to the probe.

The TOF values, expressed in microseconds, were divided by the propagation speed of ultrasound in water (equal to 1.48 mm/μs at 20 °C), computing the standoff variability in millimetres. The

Table 3
Maps of standoff between probe and scanned surface.

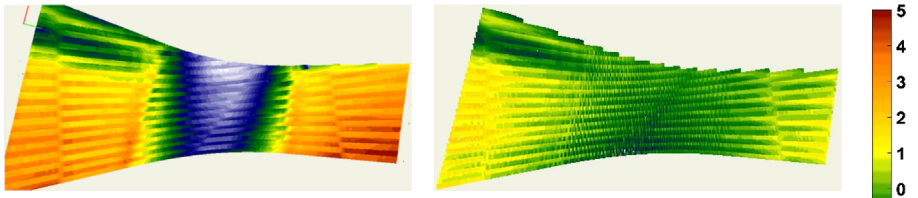
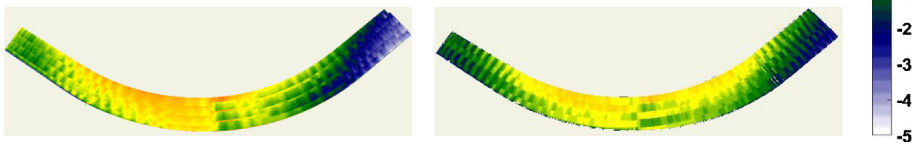
	Commercial OLP tool-paths	MFT tool-paths
Main skin		
Beam		

Table 4
Maximum, minimum, mean and Standard Deviation (SD) values of the measured standoff (all values are given in millimetres).

	Commercial OLP		MFT	
	Main skin	Beam surface	Main skin	Beam surface
Max	4.92	4.68	1.93	2.38
Min	−4.95	−4.76	−2.83	−3.30
Mean	0.83	−0.02	0.05	0.02
SD	2.33	1.69	0.79	1.17

ultrasonic energy reflected by the sample surface was sampled with a sampling frequency of 50 MHz, producing a TOF measuring resolution of 0.02 μs, thus a standoff resolution of ≈0.03 mm. The nominal programmed standoff of 35 mm was subtracted from such values, to obtain the deviations from the target standoff. The deviations relative to the RoboNDT tool-paths were compared to the deviations given by the tool-paths generated through a popular commercial OLP software application based on the Dassault Delmia V5 platform.

Whilst RoboNDT generated the tool-paths from the tessellated model of the sample, using the MFT algorithms, the commercial application used the original CAD model. The comparison was made for the robot travelling speed of 300 mm/s and acceleration of 500 mm/s^2. The data acquisition settings of the ultrasonic receiver (a Micropulse 5PA from PeakNDT (Mineo, Pierce, Wright, Cooper, & Nicholson, 2015) were configured to obtain a surface map with spatial resolution of 1.2 mm.

The experimental data demonstrated superior performance of the MFT method over the conventional OLP. The standoff distance varied within a 10 mm range for the tool-paths created with the commercial OLP software and within a 6 mm range for the RoboNDT tool-paths, generated through the MFT algorithm. Table 4 reports specific details about the standoff variability. In all cases the MFT tool-paths exhibited lower values of maximum, minimum and mean errors for both the main skin and beam surface areas. Additionally, note that the Standard Deviation (SD) values were also lower for the MFT tool-paths thus indicating lower positional dispersion around the mean values than for the conventional OLP software.

6. Conclusions

An increasing number of structures with complex shapes will need to be manufactured or maintained in coming years. The problem of generating optimum tool-paths to perform specific actions on curved surfaces through numerical control machinery or robotic manipulators will be increasingly encountered. Situations routinely arise where surface mapping to produce tessellated models is required; these being where no original CAD is available, or where parts present significant deviations from the manufacturing CAD model (often the case for large components made of composite materials). However, such tessellated models differ from precise analytical models and are not suitable to be used in current commercially available path-planning software. This work has introduced a novel Mesh Following Technique (MFT) for the generation of tool-paths directly from tessellated models without the introduction of additional approximation. It has been shown that the MFT approach can be used to find the coordinates of points on the mesh at specific distances from the edges and generate single curve trajectories as well as raster paths or more sophisticated paths (all lying on the meshed surface).

The new path-planning approach was tested through its integration into a software application developed by this paper's authors. Named RoboNDT, this application is tailored to the generation of tool-paths for the inspection of curved surfaces by 6-axis industrial robots. Comparative experiments were undertaken to evaluate the accuracy of MFT and conventional OLP tool-paths. Tool-path results were exploited to control KUKA KR16 L6-2 robots. The comparison between the commanded trajectory points and the robot feedback positions demonstrated that the MFT tool-paths were correctly generated. The variability of the standoff between the robot end-effector and the sample surfaces was monitored through an ultrasonic probe, manipulated by the robots. The deviations relative to the MFT tool-paths were compared to the deviations given by the tool-paths generated through leading commercial OLP software. The MFT tool-paths produced 40% smaller errors and up to 66% lower standard deviation values, indicating that the standoff distances were less disperse around the mean value.

The MFT algorithms were tested in the MATLAB® programming environment; however, the methods presented in this paper can also be implemented through lower lever programming languages (e.g. C# or C++) to achieve faster computing performance. In the future, MFT can support the development of versatile OLP software capable of working with tessellated surfaces. The developed method has an important role in developing robotic applications to work on objects for which the precise CAD model is not available. This is aligned with the growing use of surface mapping techniques, capable of producing tessellated models of industrial specimens and/or pieces of art.

Acknowledgements

This work has been developed in partnership with TWI Technology Centre (Wales), University of Strathclyde (Glasgow), the Prince of Wales Innovation Scholarship Scheme (POWIS) and by IntACom, a project funded by Welsh Government, TWI, Rolls-Royce, Bombardier Aerospace and GKN Aerospace. Additional support was provided with assistance from UK Research Centre in NDE (EP/F017332/1) and EPSRC Equipment Grant "New Imaging Systems for Advanced Non-Destructive Evaluation" (EP/G038627/1). The RoboNDT software, embedding the algorithms presented in this paper, is openly available from the University of Strathclyde data repository at: http://dx.doi.org/10.15129/8b38955e-8238-4f16-97c8-e58b36de8d06.

References

Andulkar, M. V., & Chiddarwar, S. S. (2015). Incremental approach for trajectory generation of spray painting robot. *Industrial Robot: An International Journal, 42*, 228–241.

Arlinghaus, S. (1994). *Practical handbook of curve fitting*. CRC Press.

Bogue, R. (2009). Finishing robots: A review of technologies and applications. *Industrial Robot: An International Journal, 36*, 6–12.

Chikofsky, E. J., & Cross, J. H. (1990). Reverse engineering and design recovery: A taxonomy. *IEEE Software, 7*, 13–17.

Choi, B., Lee, C., Hwang, J., & Jun, C. (1988). Compound surface modelling and machining. *Computer-Aided Design, 20*, 127–136.

Corke, P. (2011). *Robotics, vision and control: Fundamental algorithms in MATLAB* (vol. 73). Springer Science & Business Media.

D'Errico, J. (2010). *Polyfitn*. Available: <http://uk.mathworks.com/matlabcentral/fileexchange/34765-polyfitn>.

Fabio, R. (2003). From point cloud to surface: The modeling and visualization problem. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 34*, W10.

Freund, R. J., Wilson, W. J., & Sa, P. (2006). *Regression analysis*. Academic Press.

Gibson, I., Rosen, D. W., & Stucker, B. (2010). *Additive manufacturing technologies: Rapid prototyping to direct digital manufacturing*. New York: Springer.

Haase, W. (2013). Automated non-destructive examination of complex shapes. In *Presented at the 14th Asia-Pacific conference on NDT (APCNDT), Mumbai, India.* .

Halmshaw, R., Honeycombe, R., & Hancock, P. (1991). *Non-destructive testing*. E. Arnold.

Hassett, B., & Tschinkel, Y. (2006). Weak approximation over function fields. *Inventiones mathematicae, 163*, 171–190.

Hwang, J. (1992). Interference-free tool-path generation in the NC machining of parametric compound surfaces. *Computer-Aided Design, 24*, 667–676.

Johnson, D., & Williams, R. P. D. (1976). *Methods of experimental physics: Spectroscopy*. New York: Academic Press.

Jun, C.-S., Kim, D.-S., & Park, S. (2002). A new curve-based approach to polyhedral machining. *Computer-Aided Design, 34*, 379–389.

Martin, G. (1967). Nondestructive testing in the aerospace industry. *Ultrasonics, 5*, 274.

Mineo, C., Pierce, S. G., Nicholson, P. I., & Cooper, I. (2016). Robotic path planning for non-destructive testing – A custom MATLAB toolbox approach. *Robotics and Computer-Integrated Manufacturing, 37*, 1–12.

Mineo, C., Pierce, S., Wright, B., Cooper, I., & Nicholson, P. (2015). PAUT inspection of complex-shaped composite materials through six DOFs robotic manipulators. *Insight-Non-Destructive Testing and Condition Monitoring, 57*, 161–166.

Pan, Z., Polden, J., Larkin, N., Van Duin, S., & Norrish, J. (2012). Recent progress on programming methods for industrial robots. *Robotics and Computer-Integrated Manufacturing, 28*, 87–94.

Piegl, L., & Tiller, W. (2012). The NURBS book. *Springer Science & Business Media*.

Ren, Y., Yau, H. T., & Lee, Y.-S. (2004). Clean-up tool path generation by contraction tool method for machining complex polyhedral models. *Computers in Industry, 54*, 17–33.

Szilvśi-Nagy, M., & Matyasi, G. (2003). Analysis of STL files. *Mathematical and Computer Modelling, 38*, 945–960.

Varady, T., Martin, R. R., & Cox, J. (1997). Reverse engineering of geometric models—An introduction. *Computer-Aided Design, 29*, 255–268.

Wang, W., Zhang, Y., Scott, M. A., & Hughes, T. J. (2011). Converting an unstructured quadrilateral mesh to a standard T-spline surface. *Computational Mechanics, 48*, 477–498.

Weber, W. (2009). *Industrieroboter: Methoden der Steuerung und Regelung; mit . . . 33 Übungsaufgaben sowie einer begleitenden Internetseite: Fachbuchverl*. Leipzig im Carl-Hanser-Verlag.

Yuwen, S., Dongming, G., & Haixia, W. (2006). Iso-parametric tool path generation from triangular meshes for free-form surface machining. *The International Journal of Advanced Manufacturing Technology, 28*, 721–726.

Zahlan, N., & O'neill, J. (1989). Design and fabrication of composite components: The spring-forward phenomenon. *Composites, 20*, 77–81.