Journal of Intelligent & Robotic Systems (2022) 105:54
https://doi.org/10.1007/s10846-022-01673-6

REGULAR PAPER

Autonomous Robotic Sensing for Simultaneous Geometric and Volumetric Inspection of Free-Form Parts

Carmelo Mineo[1] · **Donatella Cerniglia**[2] · **Alastair Poole**[3]

Received: 2 February 2022 / Accepted: 5 June 2022

Abstract
Robotic sensing is used in many sectors to improve the inspection of large and/or complex parts, enhancing data acquisition speed, part coverage and inspection reliability. Several automated or semi-automated solutions have been proposed to enable the automated deployment of specific types of sensors. The trajectory to be followed by a robotic manipulator is typically obtained through the offline programmed tool paths for the inspection of a part. This method is acceptable for a part with known geometry in a well-structured and controlled environment. The part undergoing assessment needs to be precisely registered with respect to the robot reference system. It implies the need for a setup preparation phase for each new part, which can be very laborious and reliant on the human experience. This work combines real-time robot control and live sensor data to confer full autonomy to robotic sensing applications. It presents a novel framework that enables fully autonomous single-pass geometric and volumetric inspection of complex parts using one single robotised sensor. A practical and robust robot control sequence allows the autonomous correction of the sensor orientation and position to maximise the sensor signal amplitude. It is accompanied by an autonomous in-process path planning method, capable of keeping the inspection resolution uniform throughout the full extension of the free-form parts. Last but not least, a by-product of the framework is the progressive construction of the digital model of the part surface throughout the inspection process. The introduced framework is scalable and applicable to widely different fields.

Keywords Autonomous inspection · Robotic sensing · Path-planning · Data-driven control · Surface mapping · Non-destructive testing

1 Introduction

1.1 Motivation

Robotic enabled sensing has become increasingly common in recent years. Besides being used for manufacturing operations (e.g. welding, assembly, spray-painting), industrial robotic arms are also used to perform adequate inspections of safety-critical and/or high-value components. Indeed, automated non-destructive systems have attracted the interest of several industries (e.g. aerospace) to speed up the assessment of large and complex parts, overcoming the bottleneck associated with the quality assurance phases. Non-destructive testing (NDT) is the process of inspecting, testing, or evaluating parts and materials, without disrupting their functionality. Many NDT methods rely upon different physical transduction principles (e.g. sound/ultrasound, magnetism, electric field and electromagnetic radiation). The capabilities of most currently available NDT methods have been combined with some degree of automation in recent years to enhance data acquisition speed, part coverage and inspection reliability. Therefore, many automated or semi-automated inspection systems have been engineered to enable the robotic manipulation of specific types of NDT sensors. These systems have been accompanied by bespoke software applications that allow simultaneous sensor data collection and robotic positional feedback reception, merged

This work was performed at the University of Palermo.

✉ Carmelo Mineo
carmelo.mineo@icar.cnr.it

[1] Institute for High Performance Computing and Networking, National Research Council (ICAR-CNR), Palermo, Italy

[2] Department of Engineering, University of Palermo, Palermo, Italy

[3] Centre of Ultrasonic Engineering, University of Strathclyde, Glasgow G1 1XW, UK

Published online: 24 June 2022

to produce encoded sensor data maps. Automated inspection systems are usually operated by generating tool paths by offline programming software (OLP). Using the virtual model of a given part and a digital mock-up of the robotic inspection environment, OLP software can generate the required inspection tool path that follows the contour of the part surface. Then, the tool path is transferred to the robot controller and executed to control the robotic manipulation of the sensor during a given inspection. Although this approach works well when an accurate model of the part is available, and the automated inspection takes place in a well-structured environment, where the part position is precisely registered with respect to the robot reference system, it makes setting up the assessment of new parts very time-consuming and dependent on the skills and experience of the robot programmer.

Moreover, the actual geometry of a part may significantly deviate from its digital counterpart, resulting in inaccurate tool paths. For these reasons, robotic platforms should become able to flexibly scan parts through online path planning with as much autonomy as possible. Semi-autonomous or fully-autonomous inspection systems should provide the same guarantee of completeness in surface coverage achieved by a human operator manually inspecting the part or by a robot whose tool path is programmed by an expert OLP software engineer. This work introduces a new approach capable of conferring full autonomy to robotic sensing applications, providing a breakthrough in the state-of-the-art, which can be exploited in areas beyond automated NDT systems.

1.2 Related Work

Minimising the deviation between the tool paths generated through offline programming software and the contour of the actual parts is a ubiquitous problem in robotic inspections. Whereas tool paths for parts for which an accurate digital-twin model is available can be generated through commercial software with a sufficient level of accuracy, the tool paths for legacy parts or deformed components are not straightforward to generate. Some works demonstrated the possibility of obtaining good results through offline correction of the tool path deviations [1]. However, offline correction imposes a preliminary additional scanning step of the real part geometry before the automatic target operation can be carried out through the corrected path. Automated methods for free-form surface mapping have developed significantly in recent years. Photogrammetric solutions involving 3D or 2D cameras have evolved from heavily user-guided [2] to fully automated 3D model reconstruction techniques [3, 4]. Although using automated geometry reconstruction is a viable approach for some applications, it weakens the speed advantage given by robotised NDT. It is the case when running one-off inspections of custom parts or scanning composite parts, which suffer from elastic spring-back when extracted from the mould during manufacturing [5].

Moreover, geometry reconstruction requires dedicated hardware and bespoke tool paths to visit a set of target data-collection locations around the part when the 3D surface mapping instrumentation is automated through a robotic manipulator. In the manufacturing field, part geometry tolerances are usually measured using Coordinate Measuring Machines (CMMs) [6]. Their usage also relies on the user to define the sampling locations and/or a sampling spacing along a specified planar raster path [7].

The need to perform a preliminary geometry reconstruction step to inform the accurate path-planning task is removed by establishing real-time control feedback loops and enabling online corrections to the original paths. Therefore, different kinds of sensors, such as force-torque sensors [8], proximity sensors [9], laser trackers [10] and/or cameras [11, 12], have been employed to correct predefined tool paths during their execution. The usefulness to correct predefined tool paths is evident in robotic sensing applications and some specific automated manufacturing operations. An approach based on RGB-D cameras, acoustic and tactile transducers has been proposed to monitor loss of ablation and missed coverage due to misalignment in robotic grit-blasting [13]. To make automated welding more reliable and improve weld quality, the scientific literature reports methods for tool path correction based on eye-in-hand configurations through cameras [12] and laser profilers [10] for the weld seam detection.

Since automated inspection systems can be considered commonplace, the research efforts are moving towards enabling more autonomous and adaptable robotic inspections, removing the reliance on operator inputted information to guide the data-acquisition phase. Some works have investigated Bayesian optimisation and robust outlier analysis to introduce a degree of autonomy through machine learning [14, 15]. These works cast the autonomous inspection problem as an optimisation problem. After an observation is taken, a probabilistic algorithm decides where the following observation should be taken to maximise a given objective. These decisions are thus carried out sequentially as information is being gathered. If damage is detected and a sample is deemed faulty at a given step, the inspection can stop there to avoid further efforts. If no damage is present, such kind of autonomous inspection can continue focusing on minimising risk by means of exploring the space until a given target sampling density (number of samples per square surface unit) is reached. However, this probabilistic approach was only tested on small flat samples, using rectangular user-specified domains of interest. Indeed, the system autonomy is limited to within the domain of interest and assumes the presence of a signal at any visited point. Autonomy is mainly

enabled in the context of real-time signal processing, leaving the field of real-time path-planning unexplored for large and/or complex parts.

Moreover, since this approach allows any two consecutive sampling positions to be quite far from each other, with respect to the specified domain of interest, it causes an increase in robot travel time and scanning duration compared with a pre-planned inspection path covering the same area. An application field that shows some commonality with the problem at stake is the autonomous path-planning for modern vacuum cleaner robots—however, the solutions deployed in this arena work only for relatively flat parts [16]. Furthermore, although they aim to achieve entire surface covering, they do not necessarily provide consistent cleaning effort over large surfaces (the time spent on the surface unit can vary enormously).

Recently, a completely autonomous surface-profiling and part inspection process has been proposed [17]. That enables in-process surface mapping by acquiring the readings of three laser distance sensors conveniently arranged at the end-effector of a robotic arm, which is also equipped with an NDT probe for the volumetric inspection of the sample. The three distance readings are used to infer the local surface normal and the position of the surface with respect to the robot reference system. The robot position is corrected, bringing the NDT probe to a constant standoff and aligning its sampling direction to the surface normal, thus allowing a good NDT signal collection. Autonomous surface exploration is enabled through a generalised Flood Fill Algorithm (FFA) [18], therein named as Complete-Surface Finding Algorithm (CSFA) [17]. Although this system is a significant advancement in autonomous robotic sensing, it has limitations. First of all, additional sensors are required, besides the NDT inspection probe, to support the in-process surface mapping. Secondly, CSFA does not guarantee that the sampling density is kept constant over the extension of complex part surfaces and may lead to a lack of inspection coverage. Lastly, collision avoidance has not been considered.

1.3 Contribution

This work introduces a robust approach to performing a fully autonomous inspection of complex parts. It is suitable for all those situations where the target inspection is carried out through a sensor whose sensitivity depends on the probe's position with respect to the part surface. For example, this is the case for pulse-echo ultrasonic or eddy-current testing. The maximum signal amplitude is obtained when the probe is perpendicular to the test surface and at a well-controlled standoff. As a result of this work, a fully autonomous single-pass simultaneous geometric and volumetric inspection of complex parts, using only one robotised sensor, becomes possible. It has been achieved by solving

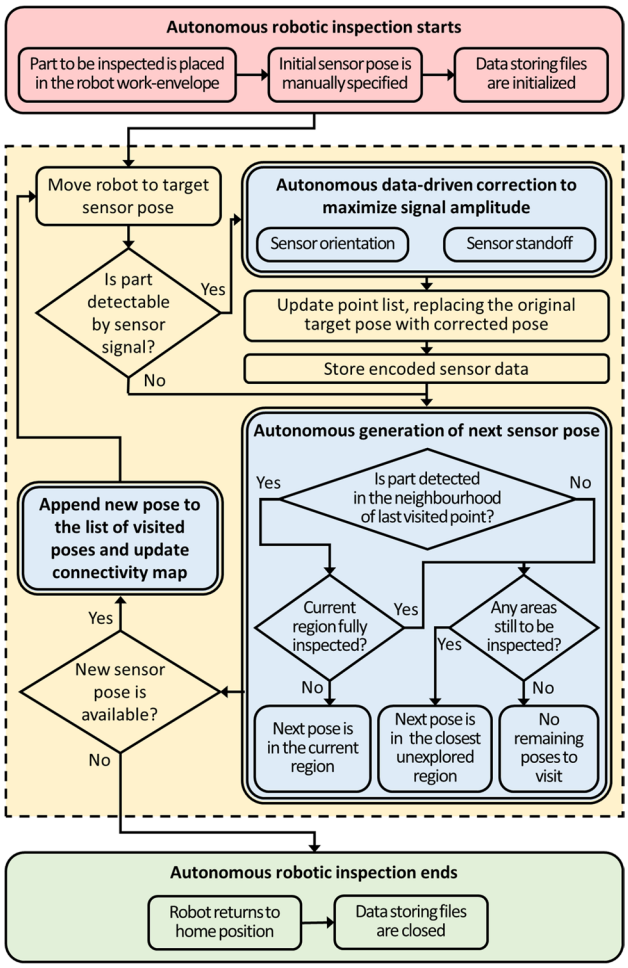

Fig. 1 High-level workflow of the system introduced by this work. The inspection autonomy is obtained through the elements within the dashed line box. The contribution of this work flowed into the areas highlighted by the blocks with a double-line border

three main problems. The contribution of this work flowed into the areas highlighted by the blocks with a double-line border in Fig. 1.

The first sub-problem consisted in developing an effective and robust algorithm and robot control sequence to enable the autonomous correction of the sensor orientation and position to maximise the signal amplitude at a given place and maintain a constant probe standoff. The second sub-problem relates to autonomous in-process data-driven path planning. This work presents a novel solution based on computing the next best point to be sampled. At the core of this is a mathematical, analytical approach that keeps the distance between the new inspection point and already visited neighbour points as close as possible to the target resolution. As a result, the inspection resolution is kept uniform throughout the full extension of the curved surface of interest. It is an important point which can lead this proposed autonomous robotic inspection system to penetrate the most

demanding industrial applications in the future. Indeed, local degradation of sampling resolution may cause some faults to remain undetected, which is unacceptable. The third subproblem concerns the tracking of the scanned areas of a part to detect the part boundaries and the autonomous sequencing of the inspected regions, minimising travel time and avoiding any risk of collisions. The proposed approach does not need a user-defined spatial domain of interest. It incrementally creates a virtual representation of the sample geometry and the surface boundary, tracking down all regions of the target surface until full coverage is reached.

Since a connectivity map of the sampled points is initiated at the beginning of the autonomous inspection and updated at each new sampled position, a virtual model of the top surface of the inspected sample volume is generated as a by-product. It can be stored in stereolithography file format (STL), which can be used for geometrically assessing the part deviations from a given digital reference. The only requirement is that the user accurately defines the starting position of the robotic inspection, allowing the surface position to be inferred through the collected sensor signal at that first position. The method enables following free-form surface geometries that cannot be globally represented analytically. Notably, it is scalable to different problem sizes, spanning from inspection of relatively small parts (e.g. through industrial robotic arms) to land surface mapping (e.g. through drones). The limitation is that the target inspection resolution cannot be smaller than the minimum curvature radius of the part surface. Therefore, in the case of significant surface discontinuities (e.g. sharp edges), the autonomous inspection system will not always be able to cross the discontinuities and continue the surface discovery on the other side of it. In practical deployments, besides being a function of the minimum defect size that one wants to detect through the inspection system, the target inspection resolution (the scanning step) must be chosen small enough to allow complete autonomous discovery of the whole surface of interest. The MATLAB-based implementation of all investigated methods is made publicly available at https://doi.org/10.5281/zenodo.5940201 and can be used by the research community for future developments.

1.4 Article Structure

The remainder of the article is structured as follows. Section 2 describes the autonomous sensor pose correction. Section 3 presents the novel approach developed to enable the autonomous in-process data-driven path planning and all additional aspects related to the supporting data structure, the inspection time minimisation, collision avoidance and stopping criteria. An application example and performance aspects are discussed in Sect. 4. Finally, Sect. 5 draws the conclusions and a prospect for future work.

2 Autonomous Sensor Pose Correction

As was said above, previous works have demonstrated the use of multiple sensors and/or machine vision cameras to address the problem of online part discovery and real-time robot pose correction. In this work, it was thought of using the data originating from the same sensor, which is used for inspecting the part, for guiding the pose correction. There is a wide range of situations where reducing the number of components in an inspection system is highly advantageous, limiting its overall cost or minimising its weight and complexity. For example, the latter is for robotic inspection systems operating in nuclear plant decommissioning or used in space exploration. The systems with higher masses are more expensive to bring into operation, and failures of complex systems may be tough to recover. Herein, the proposed solution for pose correction is composed of two phases: orientation correction and standoff correction.

2.1 Correction of Sensor Orientation

To operate a sensor with a robot manipulator, it is necessary to mount the sensor onto the robot and instruct the robot with the coordinates of the point we want to control with respect to the robot end-effector flange. In other words, it is necessary to define a tool-central-point (TCP) for the robotically manipulated sensor. The TCP is usually defined conveniently to facilitate the programming of the robot trajectory. Thus, the TCP will coincide with the focal point for optical and acoustic sensors or with a characteristic physical point for a sensor that has to contact the part to carry out the measurements.

Assuming a given sensor is brought to a target position by a robotic manipulator, with its TCP reaching the target part position but its orientation off from optimum, it is necessary to correct the sensor orientation. Regardless of the physical operating principles, the sensitivity of most transducers is maximised when they are positioned in a specific direction with respect to the normal of the part surface. For example, this is the case for laser profilers, ultrasonic sensors and eddy-current transducers. When a robotic tool-path is programmed offline, a great effort is made to ensure the robotised probe will follow the contour of a part, keeping its sensing orientation normal to the part surface. However, despite the efforts, deviations from the ideal path are often inevitable due to the differences between the virtual part model used for offline programming and the physical sample. In this work, the dependence of the sensor sensitivity on the orientation is exploited to guide the online correction of the sensing

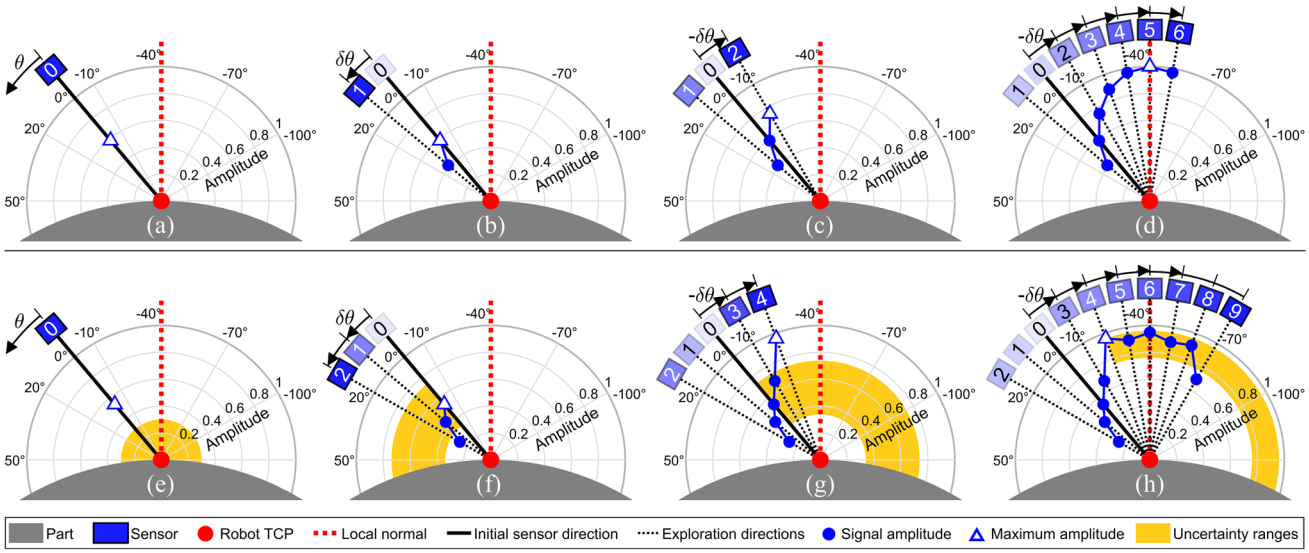

Fig. 2 Conceptual illustration of data-driven signal amplitude maximisation in the ideal case (**a-d**) and the case of lower signal-to-noise ratio (**e–h**). The initial amplitude is measured (**a, e**). The rotation direction that produces amplitude lowering is not pursued (**b, f**). The direction that gives amplitude increase is pursued (**c, g**). The rotation stops as the amplitude decreases (**d, h**)

direction. The underpinning idea consists in commanding a rotation of the sensor around its TCP whilst observing the variation of the collected signal amplitude to discover the optimum sensing direction. Whereas this concept is quite simple, implementing an effective algorithm to control the sensor rotation requires looking at the problem closer. Assuming the ideal case, where there is no sensor signal noise and starting from the bidimensional space for clarity, Figs. 2a and d illustrate the amplitude exploration process for the generic case. The figures highlight all the positions visited by the sensor, consecutively numbered and with "0" being the initial sensor position. Figure 2a shows the probe in the initial position, with the TCP lying on the surface of a curved part and the sensing direction noticeably deviating from the optimum direction. The plot is overlaid onto a polar reference system, whose origin is placed at the TCP. Whereas the radial distance from the origin represents the signal amplitude, the angular position of the sensing direction with respect to the initial direction is indicated along the circumferential axis. For the sake of illustrating the concept, the signal amplitude (a) is assumed to depend on the angle (ϑ) that the sensing direction forms with the ideal direction, as $a(\vartheta) = \cos^2\vartheta$. The mapping of amplitude versus orientation starts by rotating the probe according to the positive direction of rotation (Fig. 2b). The signal amplitude is registered at every angular increment indicated by $\delta\theta$. If such rotation direction produces amplitude lowering, the probe is brought to the opposite side of the initial sensor direction, at an angular offset equal to $-\delta\theta$, from where amplitude mapping resumes (Fig. 2c). The rotation direction that increases the amplitude is pursued until the maximum amplitude is registered and the amplitude starts decreasing again (Fig. 2d). It must be clear that, for the example illustrated in Fig. 2a-d, the negative rotation direction would not have been explored if rotating in the positive direction had resulted in an increasing amplitude. Instead, if the initial sensor direction were close enough to the optimum direction, the step illustrated by Fig. 2c would have also produced a decrease in amplitude, and the exploration would have stopped there. In the ideal case, the minimum exploration effort consists of only one rotation step in the positive direction and one in the negative direction.

In reality, sensor data are always accompanied by uncertainty due to the limited resolution of the sensors or the presence of measurement noise (e.g. electromagnetic noise) that superposes to the sensor signals. Therefore, a robust and exploitable solution for autonomous pose correction cannot neglect this fact. The subplots in Fig. 2e-h illustrate how dealing with this uncertainty is addressed in this work. These plots re-propose the same initial sensor position of the figures described previously, exemplifying what happens in the presence of noise in the sensor data. Assuming the sensor signal has an uncertainty of ± 0.1 (range width equal to $n = 0.2$), Fig. 2e shows that the exploration process can start only when the initial signal amplitude $\left(\tilde{a}_0\right)$ is higher than $1.5n$. Here, the signal amplitude is simulated adding a random component that can assume any value in the range between $\pm n/2$. The function used to generate a synthetic signal amplitude for every sensor position is: $\tilde{a}(\vartheta) = \left[a(\vartheta) + (n\Psi/2)\right]$, with Ψ being a random variable spanning between ± 1.

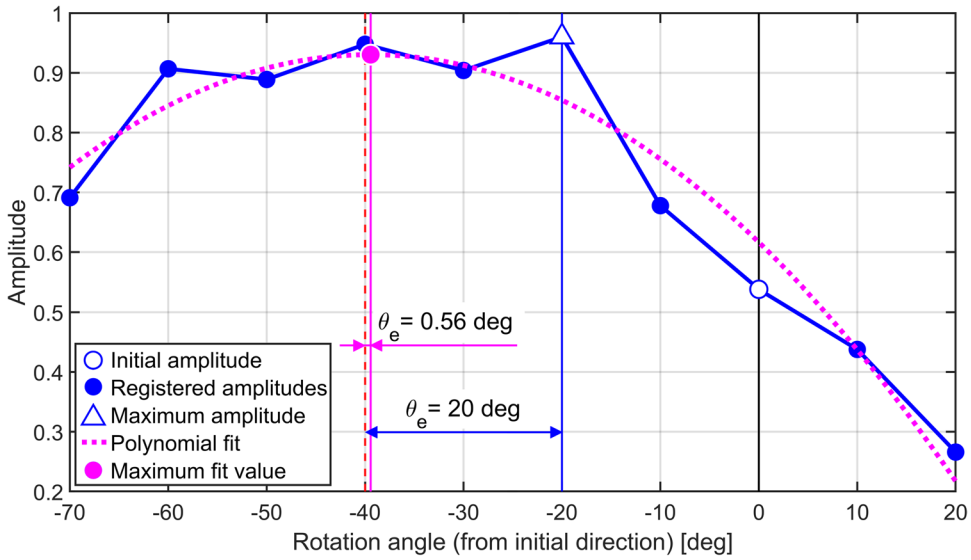

Fig. 3 Refinement of optimum sensor orientation, though polynomial fitting of registered amplitudes

In Fig. 2f, the sensor rotates in the positive rotation direction until the amplitude registered at the i^{th} step exits the uncertainty range $\left(\left(\tilde{a}_i < \left(\tilde{a}_0 - n\right)\right) \vee \left(\tilde{a}_i > \left(\tilde{a}_0 + n\right)\right)\right)$. In this example, it results $\tilde{a}_2 < \left(\tilde{a}_0 - n\right)$ and the amplitude mapping resumes at the opposite side of the initial sensor direction (Fig. 2g). Again, the sensor rotates until the registered amplitude exits the uncertainty range. In this case, it results $\tilde{a}_4 > \left(\tilde{a}_0 + n\right)$. The sensor rotation continues, until the amplitude lowers more than u, from the maximum observed amplitude (Fig. 2h). This ensures that the sensor travels across the optimum direction while mapping the signal amplitude. It must be noticed that, as it is expected, the presence of uncertainty in the data widens the total angular span that needs to be explored. Although amplitude mapping is more time-consuming, the developed strategy is effective and robust in practical applications.

The employed value of the angular sampling step ($\delta\theta$) has an important impact on the performance of the amplitude mapping process and the final correction of the sensor orientation. The value of this input parameter must be chosen consistently by the user. It is fair to say that practical values of $\delta\theta$ depend on three things: (i) the variability of the signal amplitude as a function of the angular deviation from the optimum orientation, (ii) the level of noise in the signals and (iii) the signal acquisition rate. The variation of the amplitude (versus the angular deviation) and the level of noise (n) should be considered together. The value of $\delta\theta$ should be chosen in such a way that, in the surrounding of the optimum sensing direction, the modulus of the difference between two successive amplitudes is larger than one-tenth of the noise level ($\left|a_i - a_{i-1}\right| > n/10$). From this, it results that, when $n \approx 0$, $\delta\theta$ could also be very small. However, smaller values of angular step lead to the acquisition of more signals within a given angular span. A suitable value of $\delta\theta$ should be chosen according to how much time it is acceptable to spend for amplitude mapping. Ultimately, the higher the acquisition rate provided by the available data collection instrumentation, the smaller $\delta\theta$ can be. Smaller values of angular step allow higher resolution mapping of the amplitude, increasing the probability of detecting the highest amplitude value in the surrounding of the optimum (but unknown) sensor direction. However, selecting the visited direction where the maximum amplitude is registered as the target direction may lead to significant errors due to noise in the signals. This is illustrated in Fig. 3, where the amplitude values from Fig. 2h are plotted in Cartesian axes. The direction corresponding to the maximum amplitude deviates 20 degrees from the actual optimum direction. To solve this issue, the proposed algorithm terminates with fitting a second-order polynomial curve to the sampled amplitudes. The direction corresponding to the maximum of the fitting curve is adopted as the target direction to operate the final correction of the sensor orientation. This refinement lowers the deviation to only 0.56 degrees for the example case.

In the bidimensional case, the algorithm described so far solves the autonomous sensor orientation correction. In order words, Fig. 2 illustrates the sensor moving on a single plane whilst performing amplitude mapping. Assuming such plane is perpendicular to a given unitary vector ($\vec{\boldsymbol{r}}$), the optimum sensor direction found through this planar amplitude mapping is generally only a local optimum and not the global one. Therefore, amplitude mapping must be carried out sequentially on multiple planes until no further significative amplitude enhancement is registered. Figure 4 shows the extended concept for autonomously reaching the global optimum sensor direction.

The area encapsulated by the dashed line is the core of the algorithm, where $\vec{\boldsymbol{r}}$ is chosen according to the direction

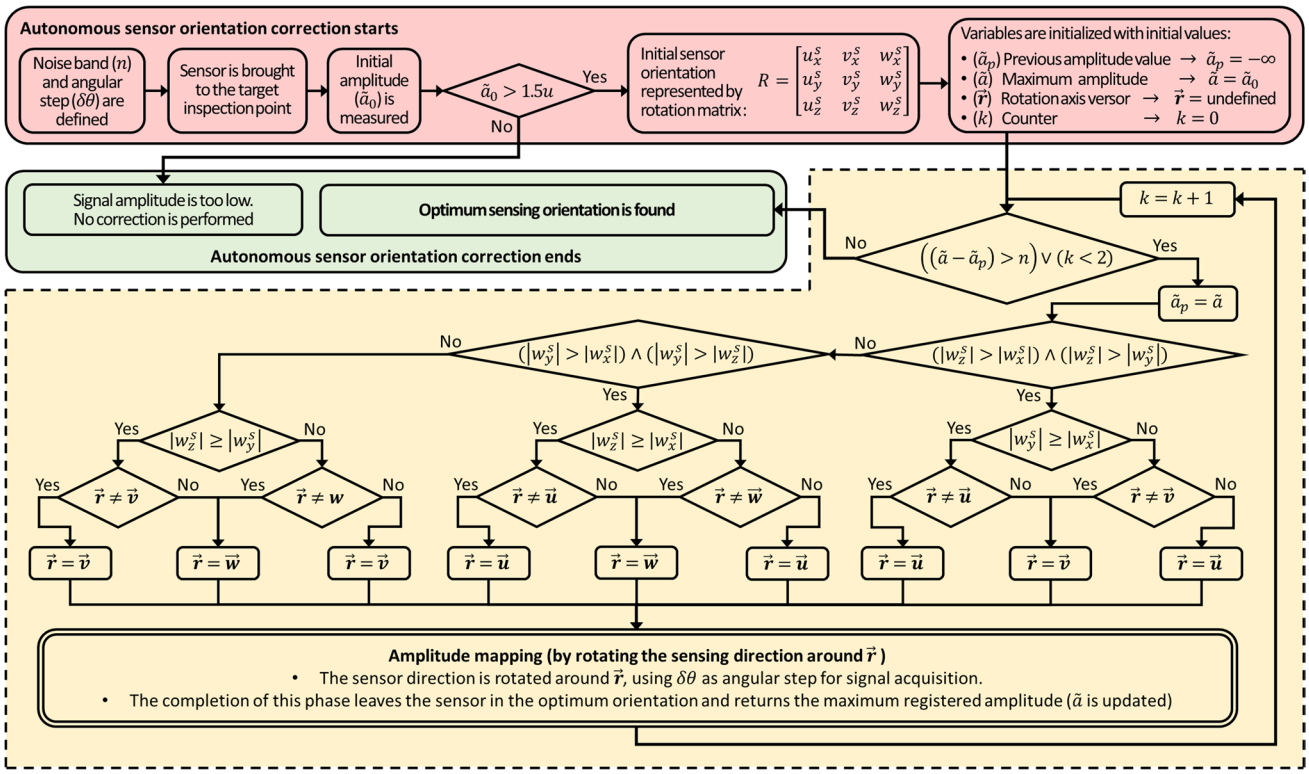

Fig. 4 Overall algorithm workflow (pseudo-code equivalent) for autonomous sensor orientation correction. The amplitude mapping is repeated multiple times, rotating the sensor around the most convenient axis until no further significative amplitude enhancement is registered

of the current sensing direction ($\vec{\boldsymbol{w}}^s = \begin{bmatrix} w_x^s & w_y^s & w_z^s \end{bmatrix}$). If the component of $\vec{\boldsymbol{w}}^s$ with the higher absolute value is the z-axis component $\left(\left(\left|w_z^s\right| > \left|w_x^s\right|\right) \wedge \left(\left|w_z^s\right| > \left|w_y^s\right|\right)\right)$, $\vec{\boldsymbol{r}}$ should either be the x-axis versor ($\vec{\boldsymbol{u}} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$) or the y-axis versor ($\vec{\boldsymbol{v}} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$). Choosing the former or the latter depends on the comparison of the other two component absolute values and on the rotation axis used at the previous iteration, which prevents rotating around the same axis twice consecutively. Likewise, if the most significant component of $\vec{\boldsymbol{w}}^s$ is the y-axis component, $\vec{\boldsymbol{r}}$ is chosen to either be the z-axis versor ($\vec{\boldsymbol{w}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$) or the x-axis versor ($\vec{\boldsymbol{u}}$). Finally, if the most significant component of $\vec{\boldsymbol{w}}^s$ is the x-axis component, $\vec{\boldsymbol{r}}$ is chosen either as the y-axis versor ($\vec{\boldsymbol{v}}$) or the z-axis versor ($\vec{\boldsymbol{w}}$). Besides checking if the updated maximum signal amplitude exceeds the previous value by more than the noise level $\left(\tilde{a} - \tilde{a}_p > n\right)$, an iteration counter ($k$) is used to make sure amplitude mapping is performed twice at the very least on two distinct planes.

2.2 Correction of Sensor Standoff

The previous section assumed the probe TCP lay on the part's surface at all times. Whereas this assumption was reasonable to focus on explaining probe orientation correction, it is not generally valid. Therefore, there is a need to correct the sensor standoff to keep the TCP on the part surfaces during the autonomous inspection. It is crucial for complex parts, whose surface curvature can cause large deviations to sensor standoff. In this work, the correction of the sensor standoff is performed after the correction of the sensor orientation. Given $\vec{\boldsymbol{w}}^s$ being the sensing direction at the end of the orientation correction, the standoff is corrected by moving the sensor TCP along such direction. The way the amount of this correction is computed depends on the type of sensor in use. For a sensor capable of measuring the distance of the part surface (e.g. a laser distance meter, typically used for surface geometry mapping), the deviation of the TCP from the part surface comes directly from the sensor reading. Other sensors can provide an indirect measure of the deviation. It is the case when distances can be inferred from the measure of an elapsed time (e.g. from time-of-flight of a wave in ultrasonic pulse-echo testing), if the wave propagation speed in the separation medium is known at a sufficient level of accuracy. Therefore, in this work, the sensor standoff is corrected through a single intervention by sending the TCP corrected target coordinates to the robotic sensor manipulator.

3 Autonomous Full-Coverage Part Exploration

To inspect the full extent of a given part, the sensor must be moved along the contour of the part surface, whilst sensor data is collected at regular spatial intervals. Automated inspection systems deploy pre-programmed tool paths, which are typically raster paths. Although these paths work well for automated systems, ensuring full part coverage with the required sampling density, they do not appear suited to autonomous inspections. To minimise the travel time while exploring an unknown part geometry, an autonomous inspection system should mimic what would be done by a blind human who is subjected to the same challenge. This work introduces an incremental data-driven inspection tool path that grows from a starting point. All aspects of the online path planning algorithm are described below.

3.1 Algorithm Supporting Data

It is helpful to define the essential data that intervenes in the algorithm's execution before attempting a description of its details. Such data consists of the scalar variables and the matrices listed in Table 1. An effort has been made to minimise the machine memory required to run this novel autonomous path planning model. The double-precision floating-point format, occupying 64 bits in computer memory, is only used to store the Cartesian and the Eulerian coordinates of the robot poses. All other data items use the 32-bit unsigned integer format when storing indices and an 8-bit Boolean variable to store logical values ("*TRUE*" or "*FALSE*"). One typical issue when running autonomous robot navigation/manipulation is that the number of locations to visit is unknown at the start, meaning that the total amount of memory space required to complete the task cannot be allocated beforehand.

Nevertheless, since memory allocation is vital to allow fast algorithm execution, all data arrays are initialised by allocating sufficient memory to store up to 100 elements, whose initial values are given in Table 1. It implies that the allocated memory needs to be enlarged whenever the pre-allocated amount is filled. Any unused allocated memory has to be released at the end.

It must also be noted that Table 1 reports only the data required for the progression of the online path planning. To keep the attention focused on the novel elements of this work, the logging of the sensor data is intentionally not discussed here. Indeed, such logging depends on the sensor data that one wants to retain and store (e.g. raw signals and/or condensed information) for real-time and/or future processing.

3.2 Initialisation from a Given Starting Pose

Given a part to inspect and the required inspection resolution (r), defined as the ideal distance that any sampling point should have from the neighbour inspection points, it is necessary to explain how the autonomous inspection commences before looking at the regime situation of the process. Figure 5 illustrates the critical steps of the start of the inspection of a generic part. It has already been anticipated that the proposed approach needs the user to specify an initial pose for the sensor to detect the part under inspection. Although that is the main requirement, some other input parameters that allow full customisation of the process commencement will be introduced and explained below. In Fig. 5a, P_1 represents the updated version of the initial user-provided pose, following the autonomous pose correction described in Sect. 2. The coordinates of this pose are stored in the first row of *pts,* and the sensor signal is acquired. If the signal amplitude is higher than $1.5n$ (with n still indicating the noise level), the first element of *isDataPt* is turned to *TRUE*. From the Eulerian angular coordinates of P_1, it is possible to compute the matrix $R_1 = \left[\vec{\boldsymbol{u}}_1^s \ \vec{\boldsymbol{v}}_1^s \ \vec{\boldsymbol{w}}_1^s \right]$, whose column vectors represent the sensor orientation in P_1.

The second sensor pose must be computed somehow for the inspection process to progress from the initial sensor pose autonomously. This pose is to be indicated with P'_2, to remember that it is a target pose to move the sensor and it can differ from the final pose (P_2, after autonomous pose correction). Since P'_2 should be at distance r from the first pose, it follows that it can be selected among the points belonging to the circumference of radius r, drawn on the plane π_1 determined by $\vec{\boldsymbol{u}}_1^s$ and $\boldsymbol{v}_1^s$ (see Fig. 5b). Indeed, using only the information acquired from the first inspection pose, such a plane is the best available approximation of the plane tangent to the part surface at P_1. It derives that the second sensor pose could be selected among the infinite points belonging to the circumference in Fig. 5b. However, to give the user a level of control over this selection, the algorithm enables the user to specify an angular parameter ($\varnothing$). This parameter indicates the user-preferred angle that the segment $\overline{P_1P'_2}$ must form with the direction originating from the projection of the x-axis versor ($\vec{\boldsymbol{u}}$) onto π_1. For the sake of illustrating an example, adopting $\varnothing = 0$, Fig. 5c shows the resultant new pose (P'_2) and its corrected version (P_2). It must be noted that the autonomous pose correction plays a fundamental role in following the curvature of the part under inspection. Indeed, whereas P'_2 inherits its Eulerian coordinates from P_1, the sensor orientation in P_2 is described by a corrected triad of versors. As soon as the robot manipulator brings the sensor to P_2, the point counters (*nPts* and *nViaPts*) are both incremented by one unit, the coordinates of P_2 are stored in the second row of *pts,* and the sensor

Table 1 Key data that intervene in the execution of the online path planning algorithm

Data name	Minimum size (rows x columns)	Format	Initial value	Description
nPts	1 × 1	Unsigned integer	1	Number of distinct visited points
pts	*nPts* × 6	Double-precision numbers	$\begin{matrix}1\\2\\\cdots\\100\end{matrix}\begin{bmatrix}x_1 & y_1 & \cdots\\0 & 0 & \cdots\\\cdots & \cdots & \cdots\\0 & 0 & \cdots\end{bmatrix}$	A matrix. Each row contains the Cartesian coordinates of one robot pose (X, Y, Z) and its Eulerian angular coordinates (A, B, C). Each distinct pose is listed only once to minimise memory consumption
isDataPt	*nPts* × 1	Boolean	$\begin{matrix}1\\2\\\cdots\\100\end{matrix}\begin{bmatrix}0\\0\\\cdots\\0\end{bmatrix}$	A Boolean value for each point TRUE (1) = sensor data has been collected at the point (the part has been detected)
isOverlapPt	*nPts* × 1	Boolean	$\begin{matrix}1\\2\\\cdots\\100\end{matrix}\begin{bmatrix}0\\0\\\cdots\\0\end{bmatrix}$	A Boolean value for each point TRUE (1) = the point overlaps with a previously inspected region
nViaPts	1 × 1	Unsigned integer	1	The number of points constituting the robotic trajectory
iViaPts	*nViaPts* × 1	Unsigned integers	$\begin{matrix}1\\2\\\cdots\\100\end{matrix}\begin{bmatrix}1\\0\\\cdots\\0\end{bmatrix}$	Indices of the trajectory points (the indices refer to the points in *pts*)
isAcquPt	*nViaPts* × 1	Boolean	$\begin{matrix}1\\2\\\cdots\\100\end{matrix}\begin{bmatrix}1\\0\\\cdots\\0\end{bmatrix}$	A Boolean value for each trajectory point TRUE (1) = sensor data acquisition attempted at the relative trajectory point
nTri	1 × 1	Unsigned integer	0	The number of triangles in the geometry reconstruction tessellated surface
iTri	*nTri* × 3	Unsigned integers	$\begin{matrix}1\\2\\\cdots\\100\end{matrix}\begin{bmatrix}0 & 0 & 0\\0 & 0 & 0\\\cdots & \cdots & \cdots\\0 & 0 & 0\end{bmatrix}$	Each row contains the indices of the vertices of one triangle. The indices refer to the points in *pts* (with no repetitions)
nEdges	1 × 1	Unsigned integer	0	The number of distinct triangle edges in the whole triangulated surface (with no repetitions)
iTriEdges	*nEdges* × 2	Unsigned integers	$\begin{matrix}1\\2\\\cdots\\100\end{matrix}\begin{bmatrix}0 & 0\\0 & 0\\\cdots & \cdots\\0 & 0\end{bmatrix}$	Each row contains the indices of the extremities of one edge. The indices refer to the points in *pts*
isOutEdge	*nEdges* × 1	Boolean	$\begin{matrix}1\\2\\\cdots\\100\end{matrix}\begin{bmatrix}0\\0\\\cdots\\0\end{bmatrix}$	A Boolean value for each edge of the triangulation TRUE (1) = the relative edge is on the perimeter of the triangulation
rDir	1 × 1	Boolean	TRUE or FALSE (specified by user)	A Boolean value to store the preferred rotation direction for the inspection trajectory TRUE (1) = clockwise; FALSE (0) = anticlockwise

signal is acquired. Thus, the second element of the trajectory index vector (*iViaPts*) is set to 2, and the second logic element of *isAcquPt* is set to *TRUE*. Finally, the sensor signal is acquired and, if the amplitude is higher than $1.5n$, the second element of *isDataPt* is turned to *TRUE*.

Once two sensor poses are visited, the third pose must be computed to allow inspection progression. Since it should be at a distance r from both the first and second pose, it follows that it should be selected among the intersections between two circumferences of radius r, respectively, centred at P_1 and P_2 and lying on the planes π_1 and π_2. However, since two circumferences drawn on different planes are not guaranteed to intersect, a more robust alternative consists in considering the intersections between the infinite cylinder (γ_1) of radius r and axis defined by $\overrightarrow{w}_1^s$ and the circumferences of radius r centred at P_2 (Fig. 5d). The problem of finding these intersections is not difficult to model using analytic geometry. First of all, both the cylinder and the circumference are translated, rotated and

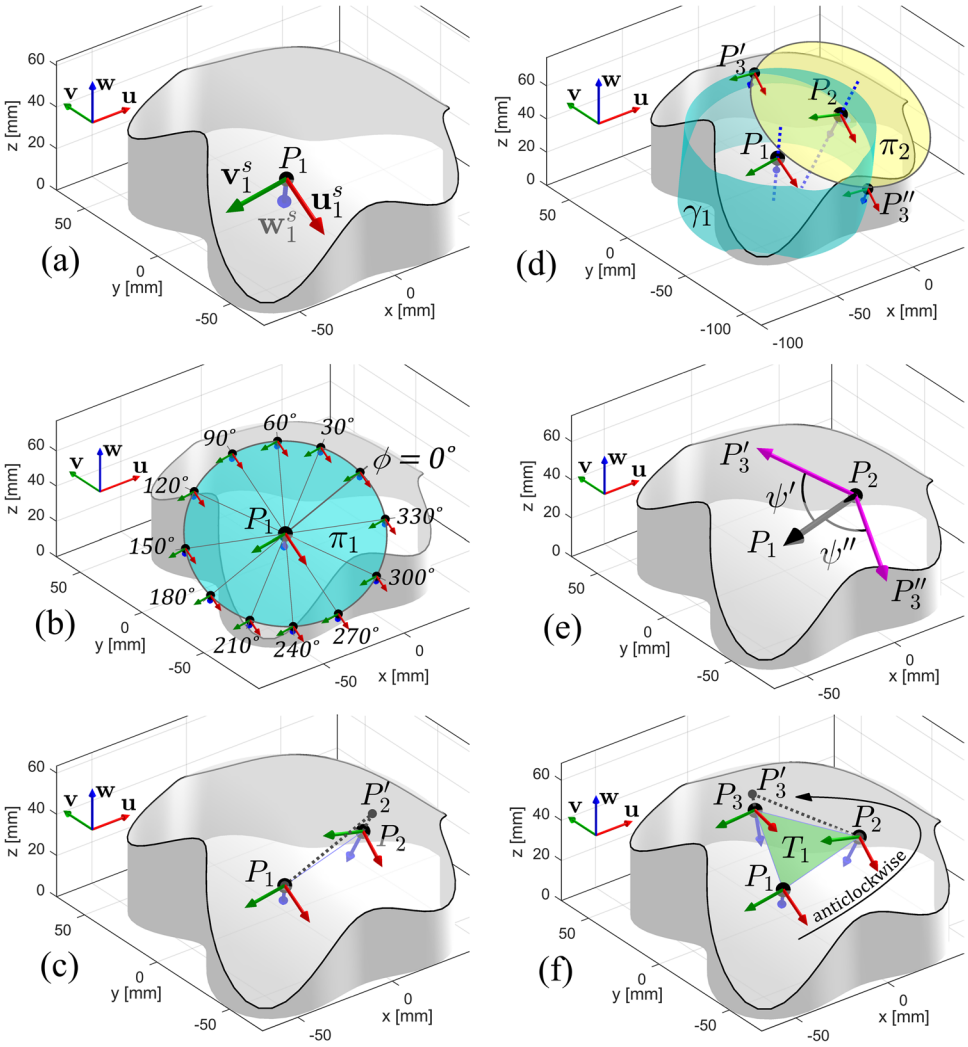

Fig. 5 Start of the autonomous inspection of a part. Example of part geometry with an initial inspection pose (**a**), computation of the second pose (**b**), correction of second pose (**c**), calculation of possible third poses (**d**), selection of the third pose according to rotation direction (**e**) and correction of third pose (**f**)

scaled to transform the γ_1 into a unitary radius cylinder centred at the z-axis. Thus, the Cartesian equation of the transformed cylinder is:

$$x^2 + y^2 = 1 \tag{1}$$

Instead, the parametric equation of the transformed circumference can be written as:

$$\begin{cases} x = \tilde{P}_x + \left[\tilde{U}_x \bullet cos(t)\right] + \left[\tilde{V}_x \bullet sin(t)\right] \\ y = \tilde{P}_y + \left[\tilde{U}_y \bullet cos(t)\right] + \left[\tilde{V}_y \bullet sin(t)\right] \end{cases} \tag{2}$$

where t is the parameter, $\tilde{P}$ is the transformed centre point of the circumference and $\tilde{U}$ and $\tilde{V}$ are the transformed orthogonal vectors in the circumference plane (originally of length r). It is possible to rationalise Eq. 2 by substituting $\cos(t) = \left(1-p^2\right)/\left(1+p^2\right)$ and $\sin(t) = 2p/\left(1+p^2\right)$, with p being a substitute parameter. Thus, combining these equations:

$$\left[\tilde{P}_x\left(1+p^2\right) + \tilde{U}_v\left(1-p^2\right) + 2p\,\tilde{V}_x\right]^2 + \left[\tilde{P}_y\left(1+p^2\right) + \tilde{U}_y\left(1-p^2\right) + 2p\,\tilde{V}_y\right]^2 = \left(1+p^2\right)^2 \tag{3}$$

Equation 3 is a quartic polynomial equation. There is no simplification of the coefficients, but the equation can be solved numerically or by closed-form formulas [19, 20]. The equation allows up to four solutions. Given the distance between the centre of the circumference and the cylinder axis being approximately equal to their common radius ($\approx r$) and the cylinder axis not parallel to the circumference plane, Eq. 3 yields only two real solutions for p. These zeros of the equation propagate into the primary parameter t and, at last, into the Cartesian coordinates of the sought intersections. The Eulerian coordinates of the two intersections are obtained from the mean of the rotation matrices of P_1 and P_2. The mean rotation matrix is computed according to the formulation proposed in [21]. Therefore, either of these two intersection points (namely: P_3' and P_3'') could be arbitrarily selected as the third inspection pose. However, in this case, to give the user control

over this selection, the algorithm enables the user to specify the initial value of the Boolean variable (*rDir*), which is used to store the preferred rotation direction of the inspection trajectory. This logic parameter supports the selection of the next sensor pose, indicating the favourite travelling direction with respect to the last travelled segment (*rDir* = *TRUE* for clockwise and *rDir* = *FALSE* for anticlockwise). Adopting *rDir* = *FALSE* for an anticlockwise travelling direction, Fig. 5f shows the new pose (P_3), originating from the uncorrected pose P'_3. As soon as the robot manipulator brings the sensor to P_3, besides incrementing the point counters (*nPts* and *nViaPts*), storing the coordinates of P_3 in the third row of *pts* and updating *isDataPt, iViaPts* and *isAcquPt*, it is possible to start constructing information about the nascent surface triangulation. Indeed, since the first three visited sensor points define the first triangle (T_1), the triangle counter (*nTri*) is set equal to 1, the triangle edge counter (*nEdges*) is set to 3, and the first three elements of *isOutEdge* are set to *TRUE*. The order with which the indices of the vertices of the first triangle and of the extremities of the first three edges are stored in *iTri* and in *iTriEdges,* respectively, depends on *rDir*:

$$\begin{aligned} &if\ rDir = TRUE \rightarrow \begin{cases} iTri(1,:) = \begin{bmatrix} 2 & 1 & 3 \end{bmatrix} \\ iTriEdges(1:3,:) = \begin{bmatrix} 2 & 1 \\ 1 & 3 \\ 3 & 2 \end{bmatrix} \end{cases} \\ &if\ rDir = FALSE \rightarrow \begin{cases} iTri(1,:) = \begin{bmatrix} 2 & 3 & 1 \end{bmatrix} \\ iTriEdges(1:3,:) = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{bmatrix} \end{cases} \end{aligned} \tag{4}$$

It must be noted that Eq. 4 lists the triangle vertices in anticlockwise order (for a viewer positioned outside of the part), regardless of the inspection travelling direction. It makes storing the triangle normal unnecessary since any software capable of importing, processing and editing 3D triangular meshes can retrieve the triangle normal from the triangle vertices. The normals will always point outward from the part triangulated surface (obeying the right-hand rule) [22].

3.3 Next-Pose Computation and Progressive Mesh Growth

The completion of the initialisation phase, through which the first three inspection poses are visited, and the first mesh triangle is created, marks the start of the main stage of the autonomous process. This phase consists of the computation of the next pose and consequent growth of the mesh through a repeating data-driven algorithm. Assuming i is the index of the current pose (P_i), $j = nTri$ and $k = nEdges$ at a given generic progress state, with T_j being the last triangle added to the mesh, the application of this algorithm allows stepping to a new pose (P_{i+1}) and the simultaneous progressive extension of the mesh. Figure 6 helps explain the algorithm, starting from three example progress states. To accompany the reading of this work, Fig. 6a-c shows the deployment of the algorithm for the case immediately following the completion of the initialisation phase described in Fig. 5. Instead, Fig. 6d-f and Fig. 6g-i relate to the other two progress states, representative of later points in time. The value of *rDir* remains set to *FALSE*, indicating that the current travelling direction is still anticlockwise. Indicating with A and B, respectively, the first and the second sensor pose that we come across when travelling from the current pose (P_i) along the external boundary of the constructed mesh, a circumference of radius r centred at P_i and lying on the plane π_i and two infinite cylinders (γ_A and γ_B) of radius r and axes $\vec{w}^s_A$ and $\vec{w}^s_B$ are constructed. A computation based on Eq. 3 is used to find the extremities of the circumference arc that remains outside both cylinders, namely: P'_{i+1} and P''_{i+1}. Thus, the extremity that produces a travel direction in agreement with the current value of *rDir* is selected as the next sensor pose. Figure 6c, f and i illustrate the updated progress state with the corrected new posture (P_{i+1}), originating from either P'_{i+1} or P''_{i+1}. This new pose allows constructing either one new mesh triangle (T_{j+1}) or two new triangles (T_{j+1} and T_{j+2}). Extending the mesh with either one or two new triangles depends on the position of the new pose with respect to A and B. In particular, only one new triangle is added, if the angle (β) formed by the vectors $\overrightarrow{AP}_{i+1}$ and $\overrightarrow{AB}$ is larger than $\pi/2$ radians (e.g. in Fig. 6c and f). Otherwise, if $\beta \leq \pi/2$ radians, two new triangles are added (e.g. in Fig. 6g). If the former is the case, the triangle counter (*nTri*) is incremented by one unit, and the triangle edge counter (*nEdges*) is incremented by two units. Whereas the elements of *isOutEdge* corresponding to the two new edges are set to *TRUE*, the element of *isOutEdge* corresponding to the edge linking A and P_i is turned to *FALSE*, since $\overline{AP}_i$ ceases to be a boundary edge of the mesh. On the other hand, if two new triangles are created, *nTri* is incremented by two units, and *nEdges* is incremented by three units. The elements of *isOutEdge* related to the edge linking P_i and P_{i+1} and to the edge linking P_{i+1} and B are set to *TRUE*. The elements of *isOutEdge* related to the edge that links B and A and to the edge that links A and P_i are turned to *FALSE*, since both $\overline{BA}$ and $\overline{AP}_i$ become internal edges of the mesh.

The order with which the indices of the vertices of the new triangle(s) and of the extremities of the new edges are appended to the respective lists, in *iTri* and *iTriEdges,* depends on *rDir* and β, according to the following generalisation of Eq. 4:

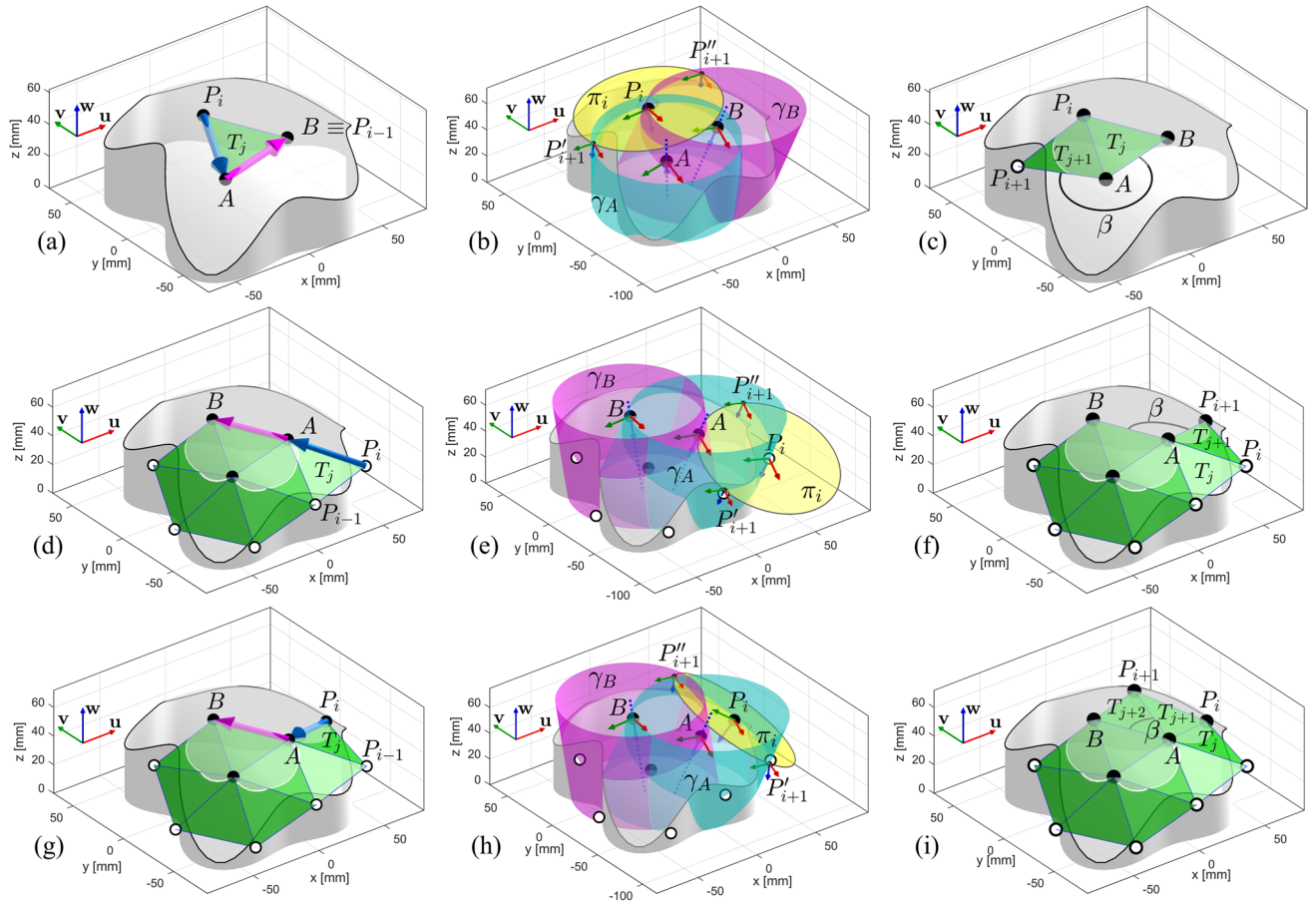

Fig. 6 Examples of next-pose computation and triangular mesh progressive growth: following the completion of the initialisation phase (**a-c**), at a later point in time (**d-f**) and at a point where two new triangles are created (**g-i**). From left to right: illustration of progress state after the visitation of the ith pose ($\boldsymbol{P_i}$) (**a, d, g**), computation of intersections ($\boldsymbol{P'_{i+1}}$ and $\boldsymbol{P''_{i+1}}$) (**b, e, h**) and illustration of progress state and mesh growth after selection and visitation and correction of the new pose ($\boldsymbol{P_{i+1}}$) (**c, f, i**)

$$
\begin{aligned}
&\text{if } \beta > \pi/2 \rightarrow \begin{cases}
\textit{if } rDir = TRUE \rightarrow \begin{cases}
iTri(j+1,:) = \begin{bmatrix} i & a & i+1 \end{bmatrix} \\
iTriEdges(k+1:k+2,:) = \begin{bmatrix} a & i+1 \\ i+1 & i \end{bmatrix}
\end{cases} \\
\textit{if } rDir = FALSE \rightarrow \begin{cases}
iTri(j+1,:) = \begin{bmatrix} i & i+1 & a \end{bmatrix} \\
iTriEdges(k+1:k+2,:) = \begin{bmatrix} i & i+1 \\ i+1 & a \end{bmatrix}
\end{cases}
\end{cases} \\
&\text{if } \beta \leq \pi/2 \rightarrow \begin{cases}
\textit{if } rDir = TRUE \rightarrow \begin{cases}
iTri(j+1:j+2,:) = \begin{bmatrix} i & a & i+1 \\ i+1 & a & b \end{bmatrix} \\
iTriEdges(k+1:k+3,:) = \begin{bmatrix} a & i+1 \\ i+1 & i \\ b & i+1 \end{bmatrix}
\end{cases} \\
\textit{if } rDir = FALSE \rightarrow \begin{cases}
iTri(j+1:j+2,:) = \begin{bmatrix} i & i+1 & a \\ i+1 & b & a \end{bmatrix} \\
iTriEdges(k+1:k+3,:) = \begin{bmatrix} i+1 & a \\ i & i+1 \\ i+1 & b \end{bmatrix}
\end{cases}
\end{cases}
\end{aligned}
\tag{5}
$$

where a and b are the indices of point A and B, respectively. It must be noted that thanks to the attention dedicated to the way the edges are listed in *iTriEdges*, the point A, required for the application of the above-described algorithm, is immediately available at any progress state. Indeed, the edge linking the last visited pose (P_i) to A is the last edge of the list. Moreover, since such an edge is always at the boundary of the mesh, there is only one other boundary edge linked to A, whose other extremity is point B. The method of indexing the robot poses, which is used in this work, minimises the consumption of computational resources to find these key points.

3.4 Inspection Confinement

There must be a way to confine the inspection into the region of interest autonomously to enable full autonomy. That is the region where the part under examination can be detected. In this work, both inspection coverage and inspection confinement are ensured by stopping the inspection

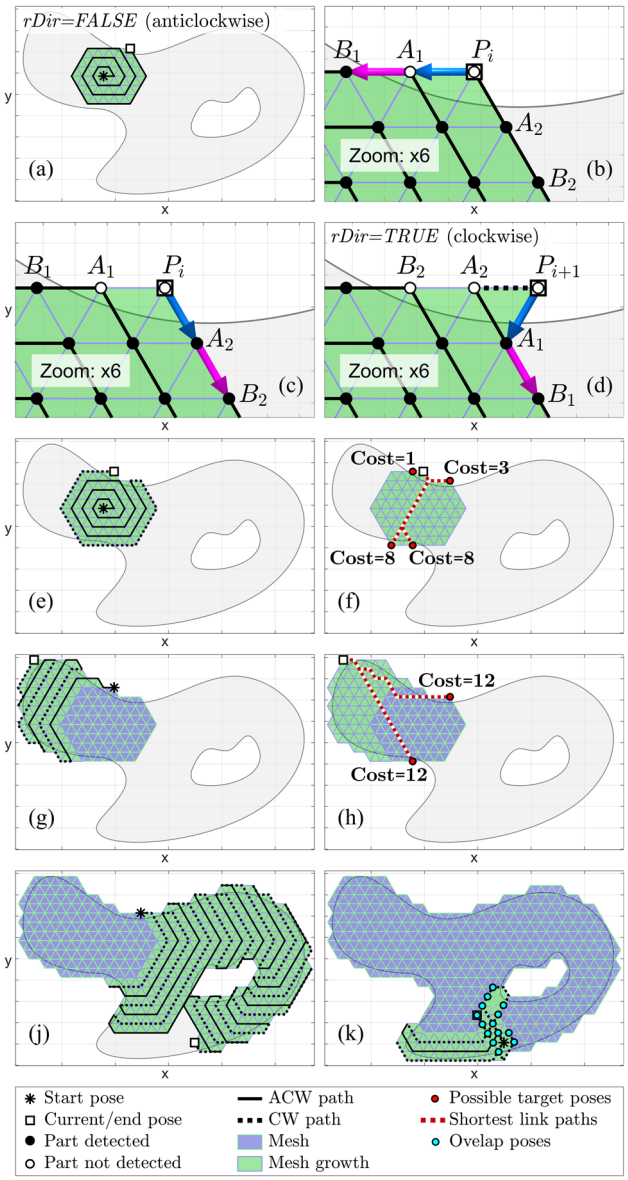

Fig. 7 Progression of inspection. The subplots show the critical steps of the process, from start to end—inspection confinement through the change of travelling direction (**a-d**), the pursuit of unexplored regions, minimising travel time and avoiding collisions (**e-j**) and the fulfilment of the stop criteria (**k**)

path from extending outside the part boundary by more than the inspection resolution (r). It is achieved by enabling changes in travelling direction in the inspection path. This concept is illustrated through the example given in Fig. 7. For clarity, the figure shows the inspection progress by looking at the process steps through a top view. The figure indicates with A_1 and B_1 the first and the second pose that we come across when travelling from the current posture (P_i) along the external boundary of the constructed mesh in the direction indicated by the current value of *rDir*. Instead, A_2 and B_2 indicate respectively the first and the second pose contiguous to P_i in the opposite direction. Given this notation, the direction of travel is changed if both P_i and A_1 are sensor poses where the part could not be detected, whereas the part could be seen in A_2 (e.g. in Fig. 7a-c). It descends that the poses A and B, used for the computation of the next pose, as described in Sect. 3.3, are taken equal to A_1 and B_1, when no change of direction is required, or equal to A_2 and B_2 otherwise. If the latter is the case, the value of the Boolean variable that indicates the current travelling direction is flipped ($rDir =\sim rDir$). Therefore, *rDir* becomes *TRUE* if it was *FALSE* and *FALSE* if it was *TRUE*, recording a change of the preferred travelling direction from anticlockwise (ACW) to clockwise (CW) or vice-versa, respectively. In Fig. 7d, the direction of travel flips from ACW to CW due to the poses P_i and A_1 being both outside the part and pose A_2 being within the part boundary. Figure 7d shows the new pose (P_{i+1}), resulting from the inverted travelling direction. Obviously, the process continues by considering the new pose as the current pose ($P_i = P_{i+1}$) and evaluating whether the part could be detected in P_i and A_1, or in A_2. It must be noted that A_1, B_1, A_2 and B_2 are always identified with respect to the current pose and the current value of *rDir*.

3.5 Collision Avoidance and Travel Time Minimisation

In more general situations, the autonomous evolution of the inspection from a given starting pose may bring to a particular progress state, where both the current pose and the two directly connected poses, which are on the external boundary of the mesh, are outside the part (e.g. in Fig. 7e and f). In this case, even though some regions of the part are still unexplored, there is no change of travelling direction that can immediately help the continuation of the inspection. It is clear that the part assessment should resume from a pose on the external boundary of the current mesh and outside the part. That pose should also be directly connected to another mesh boundary pose that falls within the part surface. Such posture would allow resuming the inspection through either the ACW or the CW travelling direction. Considering a generic case, multiple poses may meet these requirements.

Nevertheless, thanks to the indexed representation of the connectivity employed in this work, finding all suitable poses that meet the above criteria is easy and fast. Once all appropriate poses are identified, a twofold problem must be solved before the inspection can continue. First of all, it is necessary to have a criterion to guide the selection of the best pose to use. Secondly, since that pose may be quite far from the current posture, it is necessary to plan a path to move the sensor without causing any collision between the sensor itself and the explored/unexplored regions of the part under inspection.

In this work, both subproblems are simultaneously solved by employing the conceptualisation of the shortest path problem (SPP) used in graph theory [23, 24]. SPP is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimised. The current mesh is used as the graph, allowing each mesh edge to be travelled bi-directionally. The index of the current pose is given as the source, and the indexes of the identified suitable poses are provided as targets. Since all edges of the mesh have a length circa equal to the inspection resolution (r), the weight of each edge is set to be unitary, reducing the problem to an unweighted search. The A* search algorithm [25] is used to find the single-pair shortest path on the graph/mesh, linking the source to each one of the targets. The cost of each path is defined as the number of edges that need to be travelled to move from the start to the target. Thus, the target pose that can be reached through the least expensive path is selected. The relative path, originating from the solution of SPP, is used to move the sensor effectively. It is clear that, since this optimum path brings the sensor through previously visited and safe poses, it removes any possibility of collision. It must be noted that this previous statement is not necessarily true for robotic systems that can undergo mechanical singularity conditions. Travelling between two connected poses near a singularity can cause a change of robot configuration (e.g. from "shoulder up" to "shoulder down" for a six-degrees-of-freedom (6-DoF) robotic arm). However, this issue can be solved in practical implementations by storing the robot's collision-free configuration for each visited pose. Figure 7g and h illustrate the paths computed through solving the SPP to pursue the inspection of the unexplored regions of a given example part. For the progress state in Fig. 7f, the path and the target pose relative to the minimum cost (equal to 1) are selected among four possible paths. Whenever all suitable poses produce paths of equal cost (e.g. Figure 7h), the target pose is randomly chosen among them.

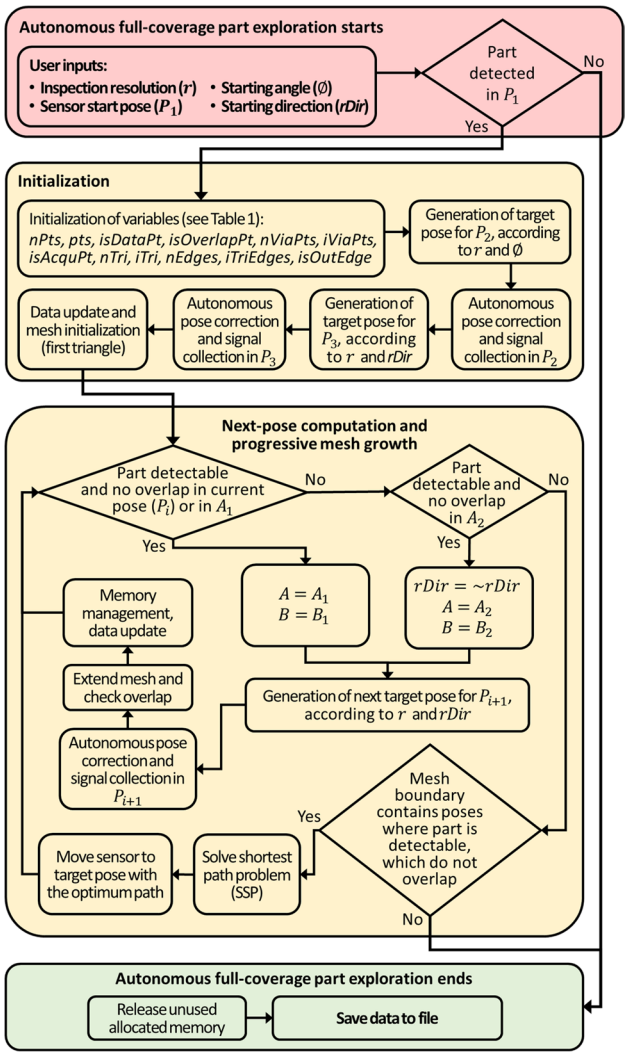

Fig. 8 Overall algorithm workflow (pseudo-code equivalent) for autonomous full-coverage part exploration

3.6 Stop Criteria

The capability of safely moving between distant poses, through solving the SPP, enables the full autonomous inspection of very complex parts.

According to the algorithm logic described, the inspection process should end when the part is not detected along the constructed mesh's boundary. It corresponds to saying that the inspection should terminate when all the elements of *isDataPt*, relative to the boundary poses, are equal to *FALSE*. However, this stopping condition may not be sufficient in some situations. The part shape may cause the inspection path to loop around a hole or obstacle and approach a previously inspected region (e.g., Fig. 7j). In this case, the inspection progression may lead to re-visiting already examined areas, resulting in unwanted prolongation of the inspection, redundant data and/or an endless inspection. To avoid this undesirable behaviour, a bespoke function of the algorithm checks if each new pose overlaps with any triangle of the constructed mesh, setting the relative element of *isOverlapPt* to either *TRUE* or *FALSE*. This function is based on the fast ray casting method presented in [26], which checks if the normal direction for the pose intersects any of the mesh triangles. Thus, all poses marked as "overlap poses" in *isOverlapPt* are considered "internal" boundary points and do not support the germination of new inspection poses. In conclusion, the autonomous inspection ends when the mesh boundary contains only poses where the part could not be detected or poses

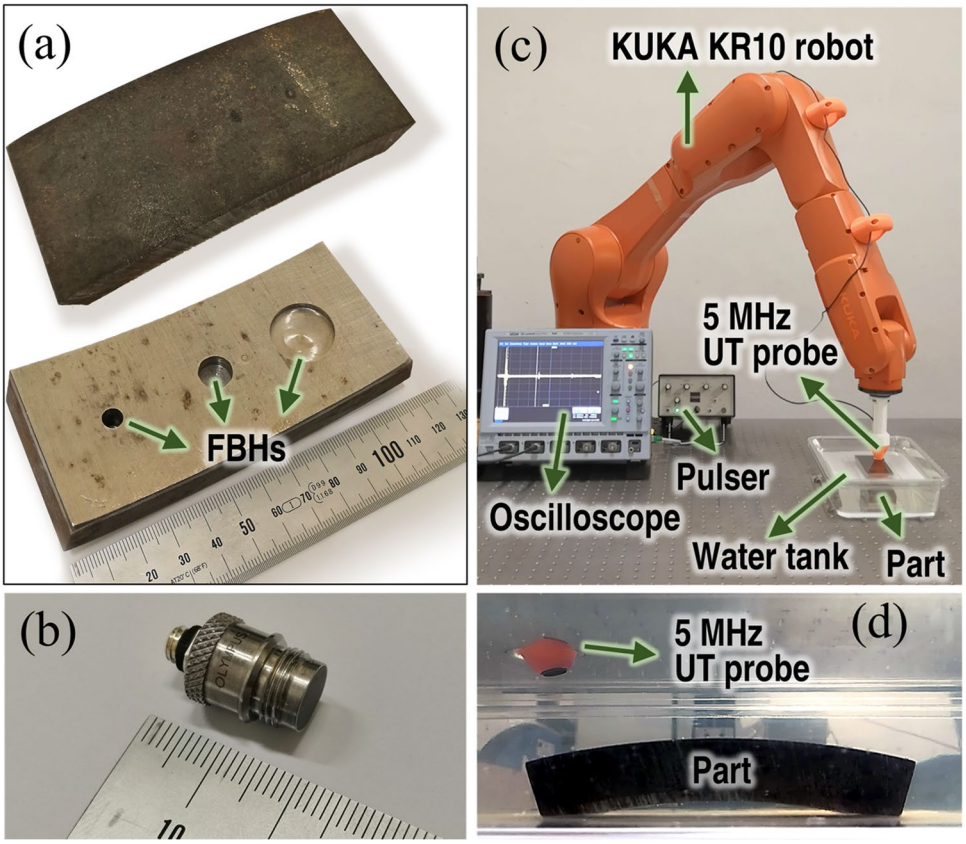

Fig. 9 Picture of top and bottom part surfaces (**a**), 5 MHz UT probe (**b**), inspection setup (**c**) and side view of part through the transparent wall of the water tank during inspection (**d**)

overlapping with boundary triangles of the mesh itself (e.g., Fig. 7k). The whole process is schematically summarised in the workflow given in Fig. 8, which illustrates the logic and uses the notation introduced above.

4 Application Example

A simple application was carried out to compare the presented autonomous inspection framework with some examples of currently established automated approaches. Since ultrasonic testing (UT) is one of the most widespread inspection methods and may be of interest to most readers, an inspection setup based on a single-element piezoelectric ultrasonic probe, manipulated by a 6-DoF robotic arm, was used. Figure 9a shows the part that was put under inspection. It is a 50 mm wide and 118 mm long sample cut out from a steel pipe with a 500 mm outer diameter and thickness of 19.2 mm. It can also be described as a 50 mm long longitudinal portion of a hollow cylinder, subtended by a circumferential angle of 27.3 deg (measured at the cylinder axis).

Three flat bottom holes (FBHs) with diameters of 6 mm, 10 mm and 20 mm were machined into the part from the concave surface to introduce artificial thickness reduction areas. The smaller diameter hole has a depth of 12 mm, the 10 mm diameter hole has a depth of 6 mm, and the largest hole has a depth of 3 mm. The UT probe used in this work (Fig. 9b) is a 5 MHz centre frequency transducer with a diameter of 6.35 mm (0.25 inches). It was mounted onto the extremity of a KUKA KR10-1100 robot manipulator through 3D-printed plastic support (Fig. 9c). The probe was used in send/receive mode (ultrasonic pulse-echo). The piezoelectric probe was excited through a pulser. The return analogue signals were digitalised with an oscilloscope at a 100 MHz sampling rate. The oscilloscope was connected to a data-collection computer during the inspection. A bespoke MATLAB-based software module retrieved the signals from the oscilloscope and encoded them with the robot's positional feedback. The computer was connected to the robot controller using the Interfacing Toolbox for Robotic Arms (ITRA) [27] to synchronise the robotic sensor manipulation with data collection. All inspections were performed through the immersion technique. Both the part and the active tip of the transducer were immersed, taking advantage of the water as a low-attenuation and stable coupling medium. The part was inspected from the convex surface, where the FBHs are not visible. Four robotic inspections were carried out to obtain full-coverage ultrasonic scans of the part. The first robotic scan was performed with a predefined Off-Line Planned (OLP) path, with the probe moving on a horizontal plane in a raster fashion and always pointing straight down into the water tank. The raster path of this first scan was

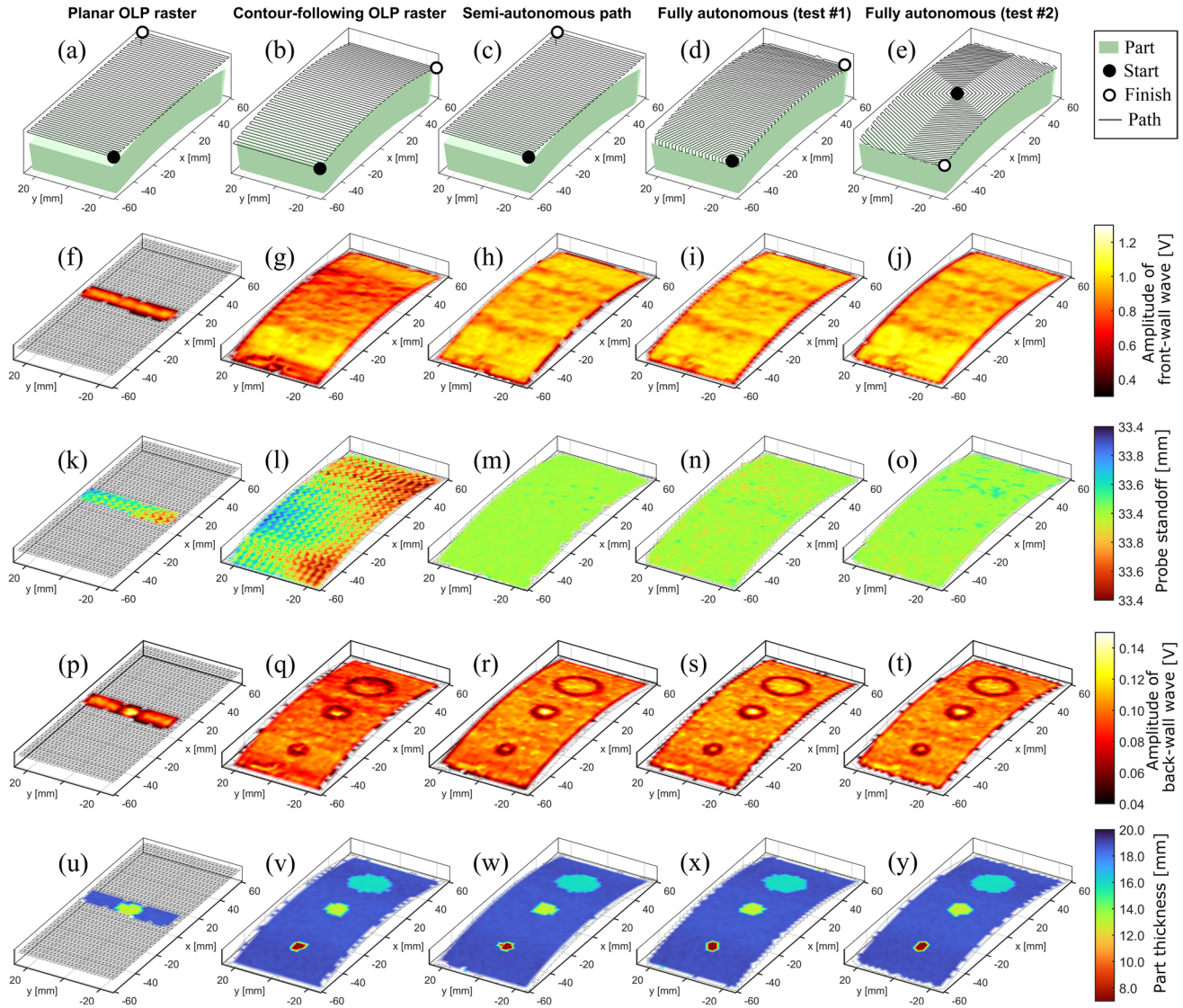

Fig. 10 Comparison of results obtained through the four different inspection types. From top to bottom: illustration of inspection paths with respect to the part geometry (**a-e**), the amplitude of front-wall UT wave (**f-j**), probe standoff (**k–o**), the amplitude of back-wall UT wave (**p–t**) and measured part thickness (**u-y**)

defined to cover a 120 mm long and 52 mm wide rectangular area centred on the part, extending 1 mm outside the part footprint at all sides. The second scan was also instructed with a predefined OLP raster path extending 1 mm outside the boundary of the part surface but following the cylindrical contour of the part. This path was generated through OLP software using the part's digital model. The normal directions to the model's surface were used to define the probe orientation at each sampling pose. The third type of scan used the planar OLP path of the first scan, but the probe orientation and standoff were corrected at each pose during the inspection process through the pose correction algorithm presented in Sect. 2. This scan is herein referred to as a *semi-autonomous* scan. Finally, the fourth scan type employed the full extent of the presented framework to obtain completely autonomous UT inspections. This fourth type was performed twice, using two different starting poses. All scans were carried out using the same inspection resolution ($r = 2$mm). The raster step was set at 2 mm, and the signal was acquired at equally spaced intervals of 2 mm for the scans using OLP paths. These fully autonomous inspections employed $r = 2$mm as target sampling resolution. Figure 10 gives a condensed illustration of the results for all scans. From top to bottom, the figure shows the employed inspection paths with respect to the part geometry, the maps of the front-wall UT wave amplitude, the maps of the probe standoff, the amplitude of the back-wall UT wave and the map of the part thickness. The probe standoff and the part thickness

Table 2 Quantitative performance parameters for all scan types

	Scanned area [mm^2]	Path length [mm]	Num. of sensor poses	Percentage of poses where the part is detected	Mean point distance [mm]	STD of point distance [mm]	STD of front-wall wave amplitude [V]	STD of probe standoff [mm]
Planar OLP	6240	3292	1647	9.22%	2.271	0.389	0.197	0.156
Contour OLP	6353	3138	1566	88.70%	2.326	0.396	0.140	0.173
Semi-auto	6124	3290	1647	87.42%	2.253	0.374	0.131	0.015
Fully auto (#1)	6297	3939	1910	90.05%	2.002	0.003	0.120	0.018
Fully auto (#2)	6269	3965	1897	90.62%	2.001	0.001	0.117	0.015

derive from the time-of-flight of the ultrasonic echo waves, taking the ultrasound propagation speed in water and steel into account. It is evident that the scan maps relative to the autonomous inspections were readily produced, applying a coloured texture to the meshes produced by the execution of the autonomous framework. For a given colour pallet, the colour given to each node of the mesh comes directly from the signal acquired at that pose, using 100% transparency for the poses where the part could not be detected. The node colours are interpolated across the surface of the mesh triangles. Mesh-based data visualisation maps were also generated for the other inspection types, using the method described in [28] to display all results in the same form and facilitate direct comparisons. Despite the gentle curvature of the part, the planar OLP raster scan was penalised by the inaccurate probe orientation. The part could only be detected for a limited portion of the scan. The probe was sufficiently perpendicular to the part surface, denoting a strong signal amplitude dependence on the alignment between the sensor sampling direction and the part surface normal. The contour-following OLP raster scan achieved full inspection coverage. However, due to the deviations between the virtual model and the real part or to the inaccurate registration of the part position, the amplitude and the probe standoff present a noticeable degree of variability with respect to the results obtained through the semi-autonomous and fully autonomous inspections enabled by this work. The deviation of the probe position from the optimum pose has a negative impact on the signal-to-noise ratio, which is evident in the map of the back-wall wave amplitude (Fig. 10q) and can reduce the capability to detect defects. It is clear that the full autonomous inspections, regardless of the starting pose used, were able to complete the scan of the part fully and produced better and more repeatable results than the human-programmed inspections.

Table 2 reports quantitative performance results. The extent of the surface area inspected through the OLP-based scans is very close to that examined by the autonomous scans. All scanned areas exceed the actual extension of the part's cylindrical surface (5956 mm^2) due to the inspection paths extending outside the part's boundary.

Interestingly, despite the similarity of all scanned areas, the fully autonomous inspections produced more sensor poses and longer inspection paths. At first sight, this may conclude that the autonomous inspections are generally slower than OLP raster paths. However, it must be observed that, in OLP raster paths, the sensor poses are arranged according to a square grid over the inspected surface. For such inspections, the user-indicated sampling resolution is only respected along the travelling direction and the stepping direction of the raster. All diagonal distances between the poses in the grid tend to exceed the target inspection resolution by a factor equal to $\sqrt{2}$. Conversely, each sensor pose is surrounded by up to six other poses with a distance similar to the target resolution (r) in fully autonomous inspections. Thanks to this reason, the sampling resolution is more uniform regardless of the directions, as denoted by the mean and the standard deviation (STD) of the distances.

5 Conclusions and Future Work

Several automated or semi-automated solutions have been proposed over the years to enable automatic deployment of specific types of sensors, speed up the inspection of large and/or complex parts and enhance inspection reliability and repeatability. Offline path-planning is typically used to instruct a robotic manipulator on the trajectory to follow for the inspection of a part. This method is only acceptable for parts with known geometry, positioned in a well-structured and controlled environment. This work presented a novel framework that enables fully autonomous single-pass geometric and volumetric inspection of complex parts using one single robotised sensor. Notably, it is scalable to different problem sizes, spanning from inspection of relatively small parts (e.g. through industrial robotic arms) to land surface mapping (e.g. through drones). An algorithm for autonomous correction of the sensor orientation and position is accompanied by an autonomous in-process path planning method. It allows keeping the inspection resolution uniform throughout the full extension of the free-form parts.

Moreover, a by-product of the framework is the progressive construction of the digital model of the part surface throughout the inspection process. The attention dedicated to the way the sensor poses are indexed in the algorithms' implementation minimises the consumption of computational resources and makes the proposed approach scalable. The framework autonomously confines the inspection into the region of interest, where the part under inspection is detectable. Full inspection coverage, collision avoidance and travel time minimisation are simultaneously solved. That is achieved by employing the conceptualisation of the shortest path problem used in graph theory. The application example highlighted that the framework works as expected, providing uniform sampling resolutions over curved part surfaces. The usage of the presented framework is not limited to a specific type of sensor and can go beyond NDT applications. Allowing autonomous and simultaneous geometric and volumetric inspection using a single robotic-manipulated sensor can play a crucial role in all those situations where reducing the number of components in the inspection system is highly advantageous. Therefore, future work should focus on testing, customising and extending the presented solutions to various scenarios.

Acknowledgements The authors thank Mr Antonino Traina and Mr Giuseppe Lo Bue, who supported the preparation of the experimental setups of this work during their university laboratory internship periods.

Author Contribution Carmelo Mineo conceived the novel elements of this work, developed and implemented the algorithms, performed data acquisition and led the writing of the article manuscript. Donatella Cerniglia supervised the findings of this work. Alastair Poole contributed to the review of the manuscript. All authors read and approved the final manuscript.

Funding This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 835846.

Availability of Data and Material The MATLAB-based source code of the algorithms described in this work is available to download at: https://doi.org/10.5281/zenodo.5940201, providing all the developed components explained in the paper.

Declarations

Ethics Approval This work did not involve human subjects and/or animals. Thus, no ethical approval was required.

Consent to Participate This work did not involve the collection of information from human subjects.

Consent to Publish This work did not involve the collection of information from human subjects.

Competing Interests The authors have no relevant financial or non-financial interests to disclose.

References

1. Zhang, H., Xu, C., Xiao, D.: Offline correction of tool path deviations for robot-assisted ultrasonic nondestructive testing. Proc. Inst. Mech. Eng. C J. Mech. Eng. Sci. **233**(8), 2879–2893 (2019)
2. Callieri, M. et al.: RoboScan: an automatic system for accurate and unattended 3D scanning. In Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004, pp. 805–812. IEEE (2004)
3. Almadhoun, R., Abduldayem, A., Taha, T., Seneviratne, L., Zweiri, Y.: Guided next best view for 3D reconstruction of large complex structures. Remote Sens. **11**(20), 2440 (2019)
4. Mineo, C., Cerniglia, D., Ricotta, V., Reitinger, B.: Autonomous 3D geometry reconstruction through robot-manipulated optical sensors. Int. J. Adv. Manuf. Technol. **116**(5), 1895–1911 (2021)
5. Mineo, C., Pierce, S.G., Nicholson, P.I., Cooper, I.: Robotic path planning for non-destructive testing–A custom MATLAB toolbox approach. Robotics Comput. Integr. Manufact. **37**, 1–12 (2016)
6. Kopáčik, A., Erdélyi, J., Kyrinovič, P.: Coordinate Measuring Systems and Machines. In: Engineering Surveys for Industry: Springer, pp. 121–141. (2020)
7. Zhang, Y., Zhou, Z., Tang, K.: Sweep scan path planning for five-axis inspection of free-form surfaces. Robotics Comput. Integr. Manuf. **49**, 335–348 (2018)
8. Mineo, C. et al.: Robotic geometric and volumetric inspection of high value and large scale aircraft wings. In: 2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace), pp. 82–86. IEEE. (2019)
9. Kamf, T., Leijon, M.: Automated mounting of pole-shoe wedges in linear wave power generators—using industrial robotics and proximity sensors. Machines **5**(1), 10 (2017)
10. Vasilev, M., et al.: Sensor-enabled multi-robot system for automated welding and in-process ultrasonic nde. Sensors **21**(15), 5077 (2021)
11. Khan, A., Mineo, C., Dobie, G., Macleod, C., Pierce, G.: Vision guided robotic inspection for parts in manufacturing and remanufacturing industry. J. Remanuf. **11**(1), 49–70 (2021)
12. Micallef, K., Fang, G., Dinham, M.: Automatic seam detection and path planning in robotic welding. In: Robotic welding, intelligence and automation, pp. 23–32. Springer (2011)
13. To, A.W.K., Paul, G., Liu, D.: A comprehensive approach to real-time fault diagnosis during automatic grit-blasting operation by autonomous industrial robots. Robotics Comput. Integr. Manuf. **49**, 13–23 (2018)
14. Gardner, P., et al.: Machine learning at the interface of structural health monitoring and non-destructive evaluation. Philos. Trans. R. Soc. A **378**(2182), 20190581 (2020)

15. Fuentes, R., et al.: Autonomous ultrasonic inspection using Bayesian optimisation and robust outlier analysis. Mech. Syst. Signal Process. **145**, 106897 (2020)
16. Hasan, K.M., Reza, K.J.: Path planning algorithm development for autonomous vacuum cleaner robots. In: 2014 International Conference on Informatics, Electronics & Vision (ICIEV), pp. 1–6. IEEE (2014)
17. Poole, A., Sutcliffe, M., Pierce, G., Gachagan, A.: A novel complete-surface-finding algorithm for online surface scanning with limited view sensors. Sensors **21**(22), 7692 (2021)
18. He, Y., Hu, T., Zeng, D.: Scan-flood fill (SCAFF): An efficient automatic precise region filling algorithm for complicated regions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (2019)
19. Shmakov, S.L.: A universal method of solving quartic equations. Int. J. Pure Appl. Math. **71**(2), 251–259 (2011)
20. Neumark, S.: Solution of cubic and quartic equations. Elsevier (2014)
21. Pariterre. "averageRT." MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/72272-averagert (accessed February 2, 2022)
22. Roscoe, L.: Stereolithography interface specification. America-3D Systems Inc. **27**(2020), 10 (1988)
23. Cherkassky, B.V., Goldberg, A.V., Radzik, T.: Shortest paths algorithms: Theory and experimental evaluation. Math. Program. **73**(2), 129–174 (1996)
24. Cormen, T.H.: Single-source shortest paths. Introduction to algorithms (2001)
25. Demyen, D., Buro, M.: Efficient triangulation-based pathfinding. Aaai **6**, 942–947 (2006)
26. Möller, T., Trumbore, B.: Fast, minimum storage ray-triangle intersection. J. Graphics Tools **2**(1), 21–28 (1997)
27. Mineo, C. et al.: Interfacing toolbox for robotic arms with real-time adaptive behavior capabilities (2019)
28. Mineo, C., Riise, J., Summan, R., MacLeod, C.N., Pierce, S.G.: Index-based triangulation method for efficient generation of large three-dimensional ultrasonic C-scans. Insight-Non-Destructive Test. Condition Monit. **60**(4), 183–189 (2018)

Carmelo Mineo received the Master's degree in Mechanical Engineering from the University of Palermo (Italy) in 2011. In 2012, he joined the Centre for Ultrasonic Engineering of the University of Strathclyde (Glasgow, UK) to undertake his doctoral studies on automated non-destructive inspection of large and complex geometries of composite materials. He became a Research Associate of the University of Strathclyde in 2015 and a Research Fellow in 2018. Carmelo was awarded a prestigious H2020 Marie-Curie Fellowship in 2020, funded by the European Commission, to lead research on Robotic Adaptive Behaviors for NDT Inspections in Dynamic Contexts at the University of Palermo. He has been a Researcher at the Institute of High-Performance Computing and Networking of the National Research Council of Italy since 2020. His current research interests comprise intelligent and autonomous robotics, advanced robot control for real-time adaptive path-planning, instrument and sensor interfacing, data collection and processing.

Donatella Cerniglia is Associate Professor in Mechanical Design and Machine Construction, at the Department of Engineering, University of Palermo, since 2015. She obtained the Ph.D. degree in Machine Construction, at the University of Palermo, and she was Visiting Scholar and Assistant Research Scientist at the Center for Nondestructive Evaluation of The Johns Hopkins University, Baltimore (USA). Her research activities, in Italy and abroad, are focused on control techniques for monitoring the integrity of structural components and robotic adaptive behaviors for NDT inspections.

Alastair Poole is currently completing an industrial PhD in TWI Technology Centre (Port Talbot, UK) in partnership with Strathclyde University. Alastair's research interests include the scanning of unknown geometries for NDT. Applying geometric knowledge from his masters in Mathematics at Durham University, his previous publications have looked at applying robotic arms to autonomous profiling, autonomous path planning and deployment over complex shapes, and force and velocity controlled scanning methodologies. His research is looking at the application of mobile robotic platforms to autonomous scanning and accurate reconstructions of complex composite structures in the context of on-site inspections.