machines

MDPI

Article

Efficient Autonomous Path Planning for Ultrasonic Non-Destructive Testing: A Graph Theory and K-Dimensional Tree Optimisation Approach

Mengyuan Zhang [1,2,*], Mark Sutcliffe [2], P. Ian Nicholson [2] and Qingping Yang [1]

[1] Department of Mechanical and Aerospace Engineering, Brunel University London, Kingston Lane, Uxbridge UB8 3PH, UK; qingping.yang@brunel.ac.uk
[2] TWI Technology Centre (Wales), Harbourside Business Park, Harbourside Rd, Port Talbot SA13 1SB, UK; mark.sutcliffe@twi.co.uk (M.S.); ian.nicholson@twi.co.uk (P.I.N.)
* Correspondence: myra.zhang@brunel.ac.uk

Abstract: Within the domain of robotic non-destructive testing (NDT) of complex structures, the existing methods typically utilise an offline robot-path-planning strategy. Commonly, for robotic inspection, this will involve full coverage of the component. An NDT probe oriented normal to the component surface is deployed in a raster scan pattern. Here, digital models are used, with the user decomposing complex structures into manageable scan path segments, while carefully avoiding obstacles and other geometric features. This is a manual process that requires a highly skilled robotic operator, often taking several hours or days to refine. This introduces several challenges to NDT, including the need for an accurate model of the component (which, for NDT inspection, is often not available), the requirement of skilled personnel, and careful consideration of both the NDT inspection method and the geometric structure of the component. This paper addresses the specific challenge of scanning complex surfaces by using an automated approach. An algorithm is presented, which is able to learn an efficient scan path by taking into account the dimensional constraints of the footprint of an ultrasonic phased-array probe (a common inspection method for NDT) and the surface geometry. The proposed solution harnesses a digital model of the component, which is decomposed into a series of connected nodes representing the NDT inspection points within the NDT process—this step utilises graph theory. The connections to other nodes are determined using nearest neighbour with KD-Tree optimisation to improve the efficiency of node traversal. This enables a trade-off between simplicity and efficiency. Next, movement restrictions are introduced to allow the robot to navigate the surface of a component in a three-dimensional space, defining obstacles as prohibited areas, explicitly. Our solution entails a two-stage planning process, as follows: a modified three-dimensional flood fill is combined with Dijkstra's shortest path algorithm. The process is repeated iteratively until the entire surface is covered. The efficiency of this proposed approach is evaluated through simulations. The technique presented in this paper provides an improved and automated method for NDT robotic inspection, reducing the requirement of skilled robotic path-planning personnel while ensuring full component coverage.

Keywords: NDT; graph theory; KD-Tree; raster scan

Citation: Zhang, M.; Sutcliffe, M.; Nicholson, P.I.; Yang, Q. Efficient Autonomous Path Planning for Ultrasonic Non-Destructive Testing: A Graph Theory and K-Dimensional Tree Optimisation Approach. *Machines* **2023**, *11*, 1059. https://doi.org/10.3390/machines11121059

Academic Editor: Dimitrios Chronopoulos

Received: 24 October 2023
Revised: 17 November 2023
Accepted: 27 November 2023
Published: 29 November 2023

1. Introduction

NDT is a critical technique used to assess the integrity of materials or structures and is widely employed in aerospace, nuclear energy, construction, and manufacturing industries [1]. With continuous advancements in society and science, intelligent robotics technology (machines with the ability to take actions and make choices) is gaining widespread application in the field of NDT. Intelligent robots offer the potential to enhance efficiency, reduce human errors, and mitigate risks through autonomous scanning and testing capabilities. Such systems are becoming increasingly feasible for NDT application due to

recent innovations in collaborative robotics [2]. Katerina recently introduced a robotic human–machine collaborative system designed to fulfil the automation requirements of NDT measurement processes within the steel production industry [3]. Canzhi, through studying dual-robot synchronous motion systems, proposed a trajectory-planning -based solution to tackle NDT challenges specific to semi-enclosed regions of complex curved composite material components. This method allows for precise trajectory planning and the successful detection of all artificial porosity defects with diameters equal to or greater than 3 mm [4]. The IntACom project, led by the TWI Technology Centre (Wales) and backed by aviation partners and the Welsh Government, has successfully developed a robotic NDT system for the rapid inspection of complex geometric composite components [5]. In addition, reference [6] presents a robot-assisted ultrasonic non-destructive testing system designed for the automated inspection of aerospace engine blades. The system effectively detects defects as small as 0.15 mm and offers high precision in thickness measurement.

In the domain of robotic NDT, path planning plays a pivotal role in achieving efficient autonomous inspection [7]. In the case of complex geometric structures, such as those found in the aerospace or automotive sector, it is usual for a highly skilled robotic and NDT operator to decompose the problem into a series of manageable inspection regions (typically a small raster region), taking care to avoid any obstacles during the planning phase, as evident from the current research methods [8]. Some efforts have been made to automate this process, utilising live feedback from vision or other sensors. For example, reference [9] proposed an innovative approach that combines force sensors, laser sensors, and RGB cameras to achieve multi-scale, collision-free robotic path planning and execution for NDT. This novel method allows for efficient path planning on noisy and incomplete point clouds generated by low-cost sensors, without relying on surface primitives. However, these approaches have some limitations with respect to the NDT probe footprint and/or geometric structure, while still requiring some level of manual optimisation. Similarly, the study in reference [10], focusing on path planning within substation environments, underscores the importance of optimising algorithms for specific contexts, particularly where the frequency and angle of turns are key evaluative metrics.

To address these challenges, a novel method is proposed for an autonomous path-planning solution based on graph-theory techniques. This approach is inspired by the algorithm introduced in reference [11], which effectively implements path planning by incorporating specific constraints in a weighted directed graph. This has provided a significant theoretical basis for our study in navigating complex environments. This solution leverages the digital model of the component and NDT probe footprint to generate paths autonomously with a greater degree of coverage, without the need for user interventions. Additionally, it addresses the need for the avoidance of obstacles and explores optimisation algorithms to improve the connectivity and traversal of nodes. In sum, this innovative autonomous path-planning solution offers a significant advancement in NDT within complex environments, setting a new benchmark for efficiency and precision.

Furthermore, the path-planning process incorporates movement restrictions, whereby the robot is limited to moving towards the nearest connected node with a preferred direction (the one with the least cost) and defines prohibited areas as obstacles. The proposed solution adopts a two-stage planning approach, initially utilising graph-based techniques and relevant concepts to determine the preferred travel directions (those that limit a sudden change in robotic movement) and subsequently employing Dijkstra's algorithm [12] to find paths to the next enclosed node, as needed.

2. Background Theory

2.1. Ultrasonic Inspection and Robotics for NDT

NDT plays a critical role in various industrial sectors, from aerospace to nuclear energy, and from healthcare to power generation. Among these, ultrasonic testing serves as an essential NDT method, employing high-frequency sound waves to detect defects in components or structures. However, manual ultrasonic inspection methods pose challenges

when applied to large components with high curvature or complex geometries. Reference [13] utilises the offline-generated scanning path technique to achieve automated eddy current non-destructive testing, specially designed for complex geometric test specimens. Furthermore, this study substantiates the exceptional performance of employing a robotic system for PAUT TWI [14], which has developed an automated ultrasonic NDT solution aimed primarily at optimising the inspection process of complex geometric components in aerospace and maritime industries by utilising robotic systems. This solution provides a platform that allows for the use of phased-array ultrasonic testing (PAUT) probes, which can be mounted to the robotic system. This offers several benefits, as PAUT allows for the electronic excitation and control of the ultrasonic signal. PAUT can swiftly sweep the ultrasonic array, meaning that robots can cover larger areas more quickly during the raster scanning process. In the context of automated robotic path planning, the important factor is the footprint of the end-effector—in this instance, the PAUT probe. These are typically larger and of different dimensions to the conventional ultrasonic probes in order to accommodate the additional elements of the array. The use of such probes allows the robot to more quickly cover the surface; however, the asymmetric footprint introduces some path-planning challenges.

Therefore, the goal of this research is to ensure that the proposed solution allows for the accuracy of ultrasonic inspection of the test components while optimising scan path planning to cover all specified areas (i.e., ensuring the probe is oriented optimally for PAUT).

2.2. Decomposition of CAD Model

A 3D CAD model typically provides a vector-based representation of a real-world object. These models are commonly reconstructed through a series of triangular meshes and are an important component in the use of robotic path planning. However, in the case of NDT inspection, a primary consideration is to ensure that the surface of a component is sufficiently scanned. This is achieved by defining NDT inspection positions along the surface of the CAD model [15]. These positions may be evenly spaced (e.g., every 5 mm based on the defect size) or more sparsely distributed for complex geometric shapes. It is also important to ensure that areas can be marked as non-inspectable, as the NDT technique may cause damage to the surface of the component or the component may cause damage to the probe. As discussed, the inspection footprint of the PAUT probe may cover a wider inspection area, thus ensuring quicker robotic scanning than would otherwise be the case.

The first challenge is to decompose the triangles that make up the CAD mesh into discrete points in the 3D space representative of the inspection points to be visited and the non-inspection points or prohibited areas. These points, being of a higher resolution than the triangles defining the surface, will ultimately form the nodes of the graph in the solution outlined within this paper. This decomposition process is well documented within the literature [16], with several approaches available. For the purpose of this work, and to avoid confusion on the specific implementations, the decomposition was performed using third party software. The CAD model was imported into MeshLab (version 2022.02), and the Poisson disk sampling algorithm was employed for random point sampling. This algorithm ensures a minimum distance between points, resulting in a more uniform distribution of the point cloud sample. This provided the set of approximately evenly distributed points along the surface of the component that were used as the input to the graph structure.

2.3. Graph Theory, KD-Tree Optimisation, and Dijkstra Algorithm

Graph theory, a pivotal discipline within mathematics and computer science, explores the intricacies of graphs [17]. These graphs are defined by vertices (or nodes) connected by edges, capturing distinct relationships between entities. Such relationships can be either directed or undirected. Graph theory is evident in various domains, including operational research, network theory, and control theory [18]. The study of the shortest path problem, a cornerstone within graph theory, has garnered extensive attention due to its relevance in practical scenarios like engineering [19].

Recognising the intrinsic benefits of graph theory for path optimisation and planning within intricate environments, we have adopted it as the foundational approach for NDT robotic path planning.

Dijkstra's algorithm stands as a seminal technique within graph theory for addressing the shortest path. It operates upon an abstract network model, where paths are conceptualised as edges and their associated weights depict parameters like distance [20]. When applied, Dijkstra's algorithm pinpoints the shortest path from a designated starting node to all of the other nodes within a weighted graph. This algorithm is particularly suitable for graphs with non-negative weights and is widely used in fields such as networks and transportation. Given these attributes, Dijkstra's algorithm provides an initial starting method for NDT robotic path planning. However, Dijkstra is primarily concerned with finding the shortest path between two or more nodes, and not an optimal strategy for NDT path planning with the number of nodes (inspection points) potentially becoming computationally prohibitive.

One method of graph theory optimisation is the use of a K-dimensional-Tree (KD-Tree) supporting structure. Here, a specialised tree data structure facilitates efficient key data searches within multi-dimensional spaces [21] (making it suitable for 3D path-planning environments). Rooted in space partitioning, a KD-Tree is similar to other spatial partitioning methodologies like Octree, Ball Tree, and Uniform Grid [22]. The KD-Tree's adaptability is its standout feature, enabling space division based on optimal search trajectories, effectively pinpointing the ideal candidate regions for complex search tasks. Its ability to accelerate searches in k-dimensional spaces has earned the KD-Tree a significant footprint in areas like cluster analysis, image matching, and information retrieval, primarily by streamlining searches and mitigating computational demands. Given its distinct advantages, our work leverages the KD-Tree as a mechanism for data structure optimisation, with the overarching goal of enhancing algorithmic efficiency.

In its implementation, the KD-Tree utilises a depth-first search (DFS) strategy, initiating binary searches from the root node and implementing backtracking when necessary [23]. This structure is not only applicable for range-based searches, but also for K- nearest-neighbour searches, swiftly identifying all of the data points within a preset distance threshold of a specific query point or locating the closest K neighbours to the query point, as shown in Figure 1. In summary, by adopting a KD-Tree for spatial partitioning and data structure optimisation, our algorithm achieves significant computational efficiency improvements in high-dimensional-path-planning tasks.

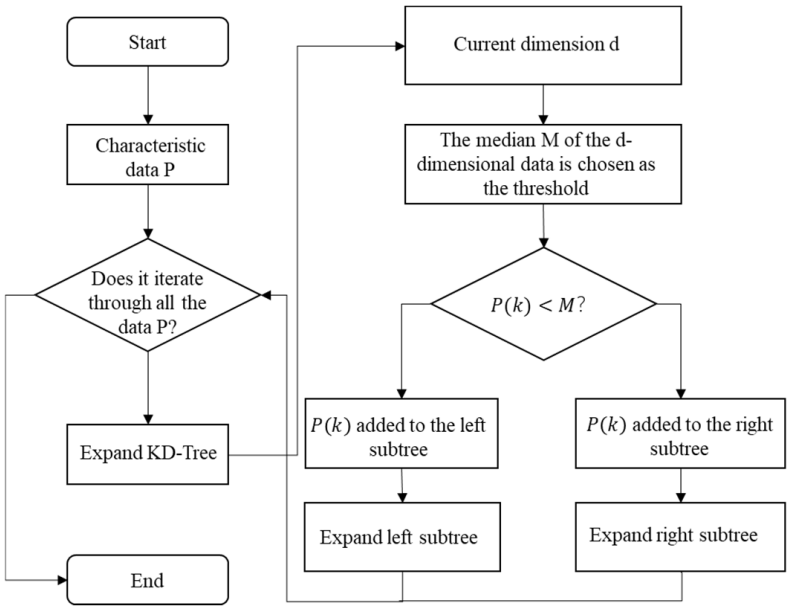

Figure 1. KD-Tree flowchart.

3. Method

While the use of graph theory provides a mechanism to represent the NDT robotic path-planning problem in 3D space, it does not in itself provide a mechanism for robotic path planning and traversal. For this, several challenges need to be overcome. Firstly, having represented the surface of a component to be inspected as a series of 3D points in space, a method is needed to optimally connect the relevant nodes (those most favourable to the robotic movement). Secondly, weights need to be computed for node connections to establish the best next movement. Thirdly, obstacles need to be accounted for (those areas prohibited for robotic movement), as does keeping track of the visited nodes. Finally, there may be scenarios where the graph traversal will become trapped (traversed into a corner). For this, a solution is needed to ensure that full surface coverage is achieved.

Our solution is to leverage graph theory, KD-Tree data structure optimisation, a novel algorithm for computing node weights, and Dijkstra's algorithm to provide an efficient NDT robotic path-planning solution in both two-dimensional and three-dimensional spaces. The following outlines our novel solution:

Initially, continuous surface data extracted from the CAD models are represented as a discrete triangular mesh. This is usually the starting point in any NDT path-planning step, with a CAD model being a digital twin of the component and environment. Using the Poisson disk sampling algorithm, these triangular meshes are converted into finer NDT inspection points. The output is a list of randomly ordered points in a 3D space, each representing an NDT inspection point. For example, the surface of a component may be several meters in size (represented as a triangular mesh within the CAD model). The NDT process (depending on the inspection requirements) may require that data are captured every 5 mm or so. The Poisson disk sampling algorithm provides a mechanism to generate these points, which will form the nodes of the graph (in this example, approximately 5 mm separation between points). An example is given in Figure 2.

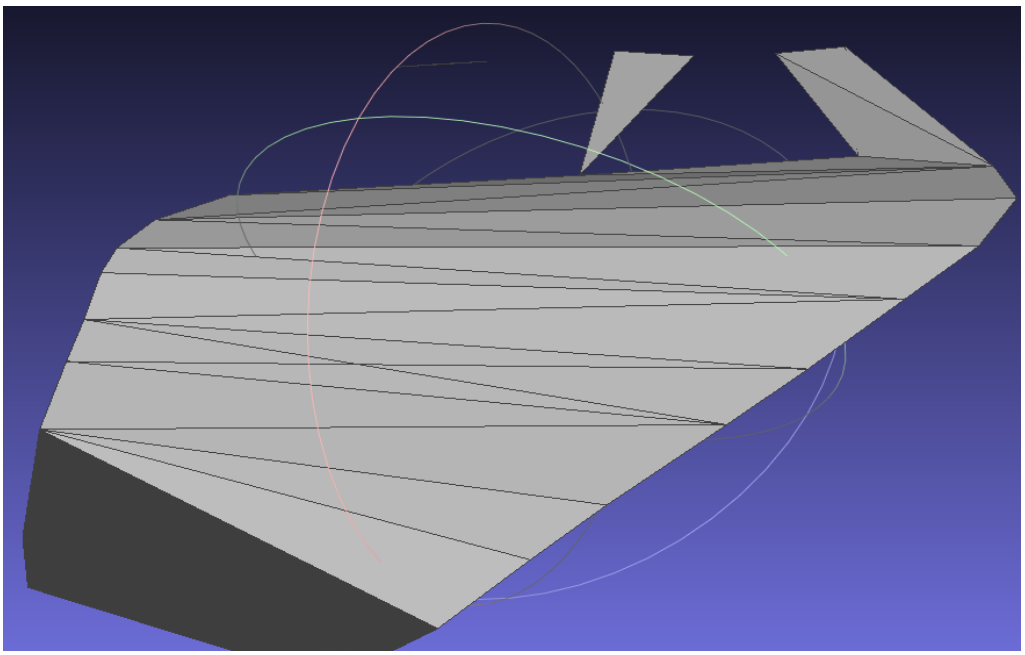

Figure 2. Visual representation of CAD triangle mesh.

Represented as vertices in the graph, G $= (V, E, W)$. Here, V is the vertex set, E is the edge set, and W is the set of weights. The vertices V denote the inspection points extracted from the CAD model, the edges E indicate the potential paths between these points, and the weights W are determined based on the actual inspection cost or time complexity based on robotic movement. In the specific implementation of this paper, the weights W represent the robotic movement priority in the following four directions: forward, backward, left, and

right. Considering the presence of restricted areas within the path-planning environment, we have designed a hybrid path-planning strategy. This is illustrated in Figure 3.

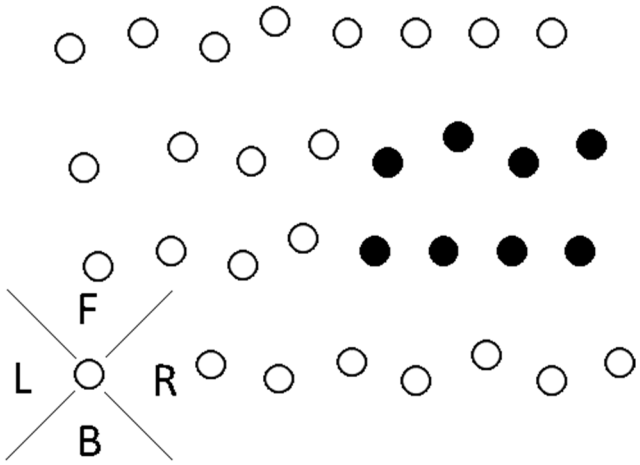

Figure 3. Hybrid path-planning strategy with directional priority weights (circle denotes graph nodes).

Represented here in a two-dimensional space for simplicity, each inspection point is an un-ordered list. Based on an initial robotic start position (the start node of the graph), KD-Tree optimisation is used to find the inspection point to the current node. This search is limited to a narrow field of view and repeated for four quadrants, such that the nearest points forward, back, left, and right are computed. These points are added to the graph as new nodes and connected directly to the current node. A weight is then computed for each node edge based on the following preferred order of movement: forward, back, left, and right.

This ensures that robotic movement has a preferred direction of forward (least travel time to next node), with back, left, and right being optional choices when no forward node is present. This results in a natural raster path as the preferred robotic movement. This concept has previously been explored by the authors through the complete-surface-finding algorithm (CSFA) [24], which incorporates flood-fill algorithms (FFAs) as a heuristic process to propagate through maps or networks, thereby discovering all positions within a connected surface or graph. Building upon this foundation, our current work introduces a modified flood-fill algorithm that enhances the process by incorporating graph theory, where localised knowledge and movement are used instead of this new approach based on graph theory. This new approach is particularly suited for un-ordered points, as it does not enforce a raster path. Instead, the algorithm computes node–node connections based on the least costly movement.

Extending Figure 3, in this example, the final graph after computation of weights is given in Figure 4. In this illustration, the green point denotes the starting point, the arrows indicate the direction of path traversal, and the grey points represent the restricted areas. As shown in Figure 4, there are some use cases where the path planning will reach a "dead-end"—meaning that all surrounding nodes are either already visited or are within restricted areas with no other eligible nodes available—whereafter the algorithm automatically switches to Dijkstra's algorithm for localised path planning. This is illustrated in Figure 4, where the red point indicates the switch. The final escape path is depicted in Figure 5. This mechanism allows the algorithm to effectively escape the "dead-end" while avoiding the restricted areas, thereby identifying the next unvisited compliant node. This is repeated recursively until all nodes have been visited.

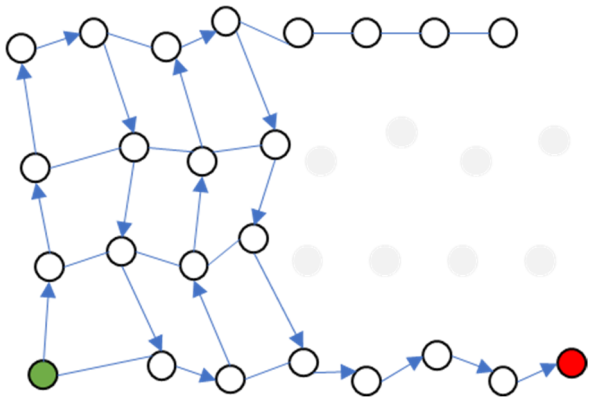

Figure 4. Resultant path planning after application of new algorithm (green denotes start and red end nodes).

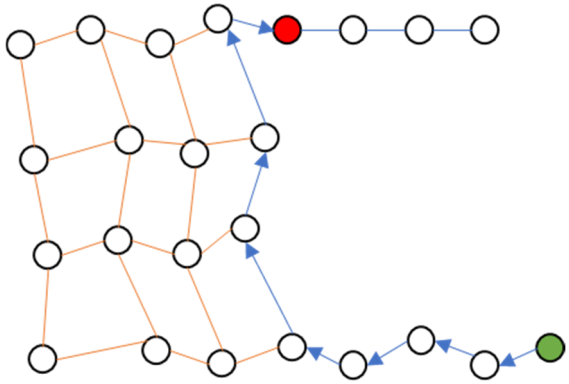

Figure 5. Escape route from "dead-end" using Dijkstra algorithm.

In this context, Dijkstra's algorithm solves for the shortest path from a source vertex v_0 to all other vertices in a weighted directed graph $\mathrm{G} = (\mathrm{V}, \mathrm{E}, \mathrm{W})$. Initially, a vertex set S is established, containing vertices for which the shortest path has already been determined—initially including only v_0. Concurrently, we maintain a distance vector dist, where $dist(w)$ signifies the current shortest path length from v_0 to w.

The complete algorithm can be summarised as follows:

1. Initialisation

Within the directed graph framework, each vertex v_i is assigned a state value, denoted as $S(v_i)$. Initially, this state value is set to positive infinity, represented by ∞, indicating that the shortest path length from the starting vertex to v_i remains undetermined. The only exception is the source vertex, which has its state value set to 0, representing a 0 distance from itself. The current node is assigned as the initial starting node—vertex v_0. This can be any node within the graph, but preferably user-selected based on the optimal starting position.

2. State Propagation

Upon evaluating a vertex v_i, it acts as a vertex of a triangle, extending towards the following four primary directions: front, back, left, and right. Consequently, four triangular regions are associated with it, each containing a set of points, as shown in Figure 2. Within each triangular extension, the algorithm sifts through the enclosed points, and, based on the Euclidean distance coupled with directional priority, it selects an optimal adjacent point. This strategy ensures that the adjacent points are chosen not only based on distance, but also on directional preference, ensuring path continuity and maximising spatial utilisation.

3. Data Structure Optimisation

Considering the complexity of navigation in a three-dimensional space and potential data scalability challenges, the adoption of the KD-Tree data structure becomes imperative. Storing vertices from the graph within a KD-Tree ensures an efficient nearest-neighbour query within a time complexity of O(logn), thereby significantly improving the algorithm's performance efficiency.

4. State Update

Once the least costly path from the current node to the connected nodes is determined, the current node is updated to this new position and the previous node marked as visited—v_i is marked as processed. Furthermore, v_i is incorporated into the set S, which contains all vertices for which the least costly path has been determined. This is repeated until the current node has no available unvisited connections (i.e., a dead-end state encountered).

5. Dead-end Escape

A "dead-end" is met when all proximal points in the four primary directions from a vertex have either been traversed, fallen within an obstacle, or aligned with the model's edge. Dijkstra's algorithm is used to allow the probe to escape based on the principle of the shortest path to the nearest unvisited nodes traversing through the graph. Then, we return to Step 4 and repeat until all nodes are visited.

The pseudocode of the hybrid algorithm used in this study is presented in Algorithm 1.

Algorithm 1: The pseudocode of the hybrid algorithm

```
Function calculate_distance(nodeA, nodeB):
    return sqrt((x_b − x_a)^2 + (y_b − y_a)^2) // Euclidean distance

Function find_nearest_unvisited_node(n_current):
    n_unvisited = {n_i ∈ Node s | n_i.visited = false} // Set of unvisited nodes
    return argmin_{n ∈ n_unvisited} calculate_distance(n_current, n)

Function update_robot_footprint(n_current):
    Nodes within the range of robot width from n_current -> visited = true
Function move_robot(n_current, direction):
    n_current = n_current.direction if n_current.direction != null and
n_current.direction.visited = false

Function find_path(n_start, n_target):
    for each n ∈ Nodes:
        n.tested = false
        n.w = infinity
    n_start.w = 0
    Queue = {n_start}
    while Queue != empty:
        n_current = dequeue(Queue)
        n_current.tested = true
        for each n_adjacent ∈ {n_current.forward, n_current.backward, n_current.left,
n_current.right}:
            dist = n_current.w + calculate_distance(n_current, n_adjacent)
            if dist < n_adjacent.w:
                n_adjacent.w = dist
                if n_adjacent.tested = false:
                    enqueue(Queue, n_adjacent)

    Path = empty stack
    n_current = n_target
    push(Path, n_current)
```

Algorithm 1: *Cont.*

```
    while n_current ! = n_start:
        n_min = argmin_{n ∈ {n_current.forward, n_current.backward, n_current.left,
n_current.right}} n.w
        n_current = n_min
        push(Path, n_current)
    return Path
Main program:
    while there exist n ∈ Nodes such that n.visited = false:
        n_current = find_nearest_unvisited_node(n_current)
        if n_current exists:
            update_robot_footprint(n_current)
        move_robot(n_current, direction) // direction ∈ {forward, backward, left, right}
        find_path(n_current, n_target)
```

At initialisation, if there is an edge from v_0 to w, then $dist(w)$ is the weight of the edge. If there is no path from v_0 to w, then $dist(w)$ is set to infinity.

During the iterative process of the algorithm, a vertex u is chosen where $dist(u)$ is the minimum among all vertices not in set S, as follows:

$$dist(u) = \min\{dist(w) | w \notin S, w \in V(G)\} \tag{1}$$

This vertex u is then added to S. At this time, $dist(u)$ is the shortest path length from v_0 to u. Moreover, for all vertices w not in S, if a shorter path can be obtained through the newly added vertex u to S, we carry out the following:

$$dist(u) + cost(u, w) < dist(w) \tag{2}$$

then update as follows:

$$dist(w) = cost(u, w) + dist(u) \tag{3}$$

Once all vertices are added to S, the algorithm ends, and the dist value of each vertex v is the shortest path length from v_0 to v. For completeness, the full Dijkstra algorithm applied in this paper is shown in Algorithm 2.

Algorithm 2: Application of Dijkstra's Algorithm for Determining Shortest Path Lengths in a Graph

```
function Dijkstra(G, v0):
    // Initialize distance array and set of vertices S
    dist[] = {infinity} // An array to store the shortest distance from v0 to each vertex
    dist[v0] = 0
    S = empty set    while S does not contain all vertices in G:
        // Find the vertex u with minimum dist value and add it to S
        minDist = infinity
        u = None
        for each vertex v in G:
            if v not in S and dist[v] < minDist:
                minDist = dist[v]
                u = v
        add u to S
        // Update the shortest distance to other vertices through u
        for each neighbor w of u:
            if w not in S:
                newDist = dist[u] + cost(u, w)
                dist[w] = min(dist[w], newDist)

    return dist
```

4. Simulation and Results

In the simulation configuration, a two-dimensional sample space was computationally generated, structured as a grid with discrete vertices, denoted as v_i, at intervals of six units. Each v_i encapsulates explicit x and y coordinate values, representing its spatial position within the bi-dimensional domain. Additionally, within this sample space, a specified area with x and y coordinates ranging between 15 and 50 is explicitly demarcated as a prohibited area. This area serves as a simulation mechanism to represent physical obstacles or impassable regions within the sample space. In Figure 6, the red region denotes the coverage area of the probe, with a width of three units. The blue dots signify the discrete vertices v_i within the sample space grid.

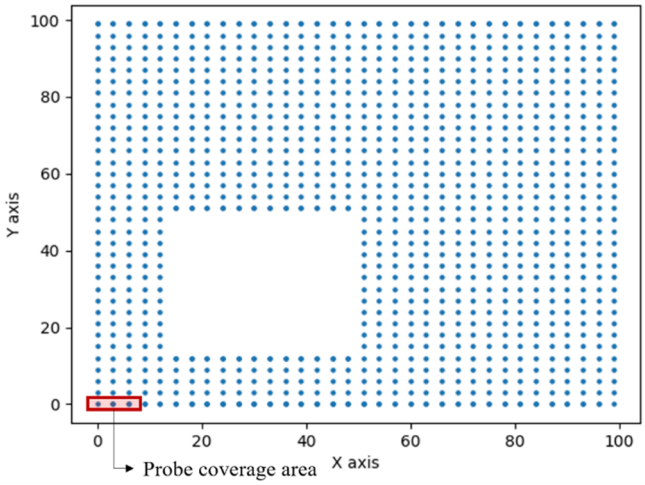

Figure 6. Two -dimensional sample space visualisation with prohibited zones.

Setting off from the coordinate (0, 0), the algorithm operates according to its predetermined logic to identify the next feasible node, whilst steering clear of the prohibited zones. It proceeds in this fashion until it has traversed all of the nodes within the sample, thereby formulating an optimal path for probe inspection. The global path is indicated by the red line with an arrow in Figure 7. The resultant route embodies the algorithm's efficacy in manoeuvring through a constrained environment, and it is potentially the most efficient trajectory under the given parameters.

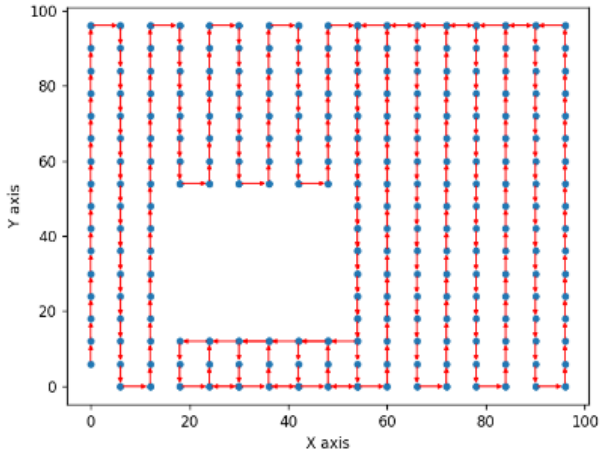

Figure 7. Results of the 2D sample pathfinding.

In order to assess the reliability and generalisation capability of the hybrid algorithm, we devised a series of diverse sample scenarios for validation. Figure 8 shows a distinct L-shaped domain. This configuration is bifurcated into the following two segments: the

first is a vertically oriented rectangle with dimensions of 60 units in width and 240 units in height, originating at the coordinates (40, 40); the second is a horizontally oriented rectangle measuring 200 units in width and 60 units in height, spanning from (40, 40) to (240, 100). Within this L-shape, an obstacle of 30 units in width and 40 units in height is positioned, ranging from (50, 80) to (80, 120). Figure 9 presents an inverted "T" configuration. The main body is a vertically aligned rectangle of 80 units in width and 240 units in height, with starting coordinates of (100, 40). Atop this, a horizontal rectangle extends, sized at 200 units in width and 60 units in height, ranging from (40, 40) to (240, 100). An internal obstacle, measuring 40 units in width and 70 units in height, is located between coordinates (120, 80) and (160, 150). Figure 10 shows a specimen reminiscent of a "rectangular ring". This construct predominantly comprises the following two elements: the external perimeter is a rectangle of 200 units in width and 240 units in height, with its origin at (40, 40), while the internal component is a rectangle of 80 units in width and 160 units in height, commencing at (100, 70). Within this "rectangular ring," another obstacle, spanning 40 units in width and 70 units in height, occupies the region from (100, 10) to (150, 50). Through validation across various two-dimensional geometric samples, the hybrid algorithm demonstrates pronounced reliability in the bi-dimensional space. The algorithm not only adapts efficiently to a myriad of geometric scenarios, but also adeptly avoids obstacles, ensuring pathway integrity and superior optimisation, whilst navigating clear of a dead-end.

In order to more profoundly and intuitively elucidate the advantages of the application of the KD-Tree in data processing, this study tabulates the data processing speeds when employing the KD-Tree data structure versus those without it, as shown in Table 1. Furthermore, to quantify the performance discrepancy between the two, we have also calculated the performance enhancement rate.

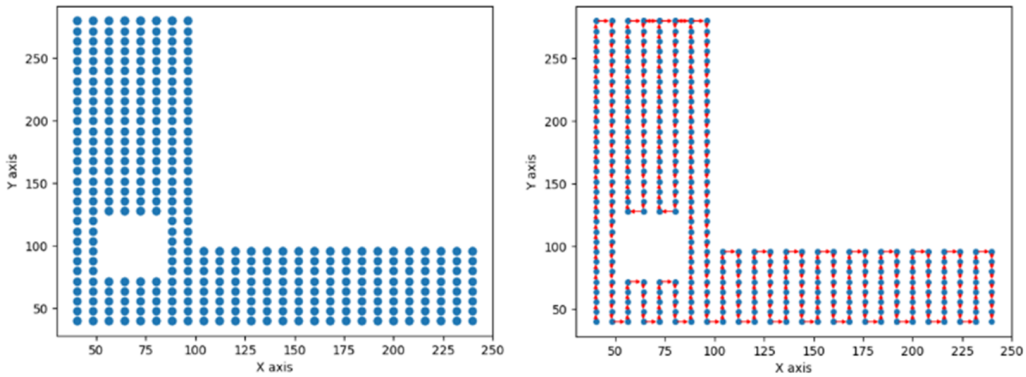

Figure 8. Path planning result of L-shaped sample.

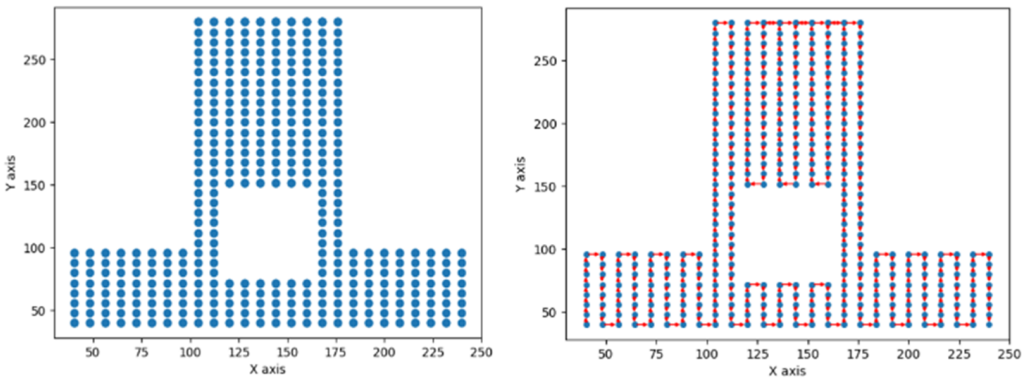

Figure 9. Path planning result of inverted "T" sample.

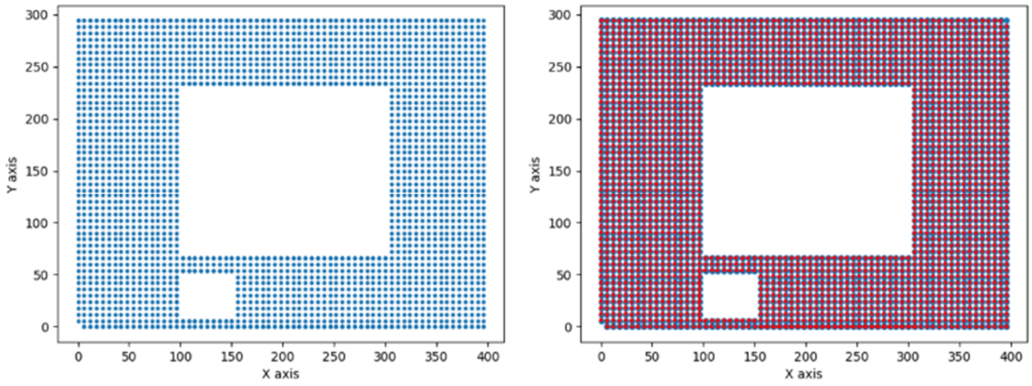

Figure 10. Path planning result of "rectangular ring" sample.

In order to further assess the performance of the algorithm, this study tested it on a 3D point cloud data model of a car door. This model was processed using the open-source software MeshLab and transformed into a point cloud representation, as shown in Figure 11, consisting of 9709 sampled points. The path result of the 3D data processed through a hybrid algorithm is shown in Figure 12, where the coordinate system is represented red, green, and blue arrows corresponding to the X, Y, and Z axes, respectively.

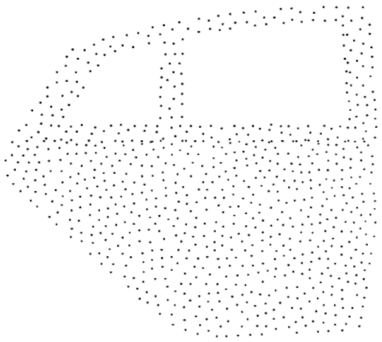

Figure 11. Three-dimensional point cloud model of the car door.

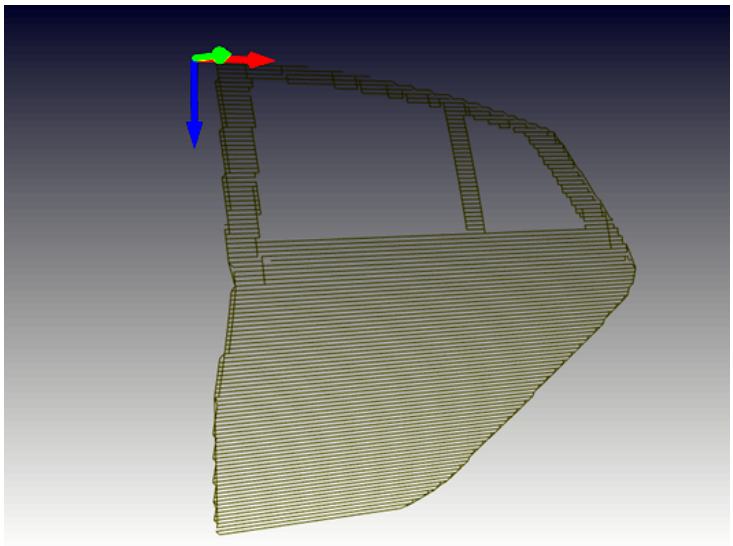

Figure 12. The scan path result processed from the 3D data.

Table 1. Performance comparison between KD-Tree and Non-KD-Tree.

X	Y	KD-Tree Time (ms)	Non-KD-Tree Time (ms)	Performance Improvement
500	800	336	3376	90.04%
800	1000	1380	10,907	87.34%
1200	1400	3000	45,251	93.37%
1600	1800	5857	134,769	95.64%
2400	2600	19,002	790,297	97.60%
3200	3400	50,162	980,996	94.89%
4000	4200	95,388	2,570,809	96.29%

5. Conclusions

In the realm of NDT robotic scanning of intricate structures, offline methods necessitate precise digital models and user intervention, posing challenges linked to model accuracy, skilled personnel, and integrating both the NDT inspection modality and the component's geometric intricacies. Addressing efficient complex surface coverage, we introduce a scanning technique using a hybrid algorithm approach tailored to the dimensional constraints of ultrasonic phased-array probes, with scan rasterization during robot movement. By synergising computer-aided design (CAD) principles, we advocate for an autonomous path generation method devoid of direct user involvement. Our methodology incorporates ultrasonic inspection system considerations, particularly phased-array transducers, and harnesses graph theory and KD-Tree optimisation, striking a balance between straightforwardness and efficiency. To facilitate robot navigation, explicit prohibitive areas are identified as obstacles. We propose a dual-stage planning model intertwining a modified flood-fill algorithm with Dijkstra's algorithm, ensuring continuity in the presence of movement constraints. The entire process is cyclic, persisting until comprehensive surface coverage is attained. Simulative evaluations validate our method's efficacy, underscoring the potential of amalgamating graph-theory principles, KD-Tree optimisation, and Dijkstra's algorithm in pioneering autonomous ultrasonic scanning path planning.

This paper sets the stage for future research that will delve deeper into human–robot interactions within our autonomous path planning in NDT, acknowledging the complexities of such collaborations. Additionally, we will conduct further experiments to validate our algorithm's efficiency and assess its practical robustness and applicability in varied real-world NDT contexts. This approach aims to ensure that our methodology not only excels in theoretical development, but also proves its practical worth in industrial settings.

Author Contributions: Conceptualisation and Methodology, M.Z. and M.S.; Writing—original draft preparation, M.Z.; Supervision, M.S., P.I.N. and Q.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This project was part of an initiative known as AEMRI (Advanced Engineering Materials Research Institute), which is funded by the Welsh European Funding Office (WEFO) using European Regional Development Funds (ERDF) WEFO contract no. 80854.

Data Availability Statement: Data are contained within the article.

Acknowledgments: This work was enabled through the National Structural Integrity Research Centre (NSIRC), a postgraduate engineering facility for industry-led research into structural integrity established and managed by TWI Ltd. through a network of both national and international universities.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dwivedi, S.K.; Vishwakarma, M.; Soni, A. Advances and Researches on Non Destructive Testing: A Review. *Mater. Today Proc.* **2018**, *5*, 3690–3698. [CrossRef]
2. Bogue, R. The Role of Robotics in Non-destructive Testing. *Ind. Robot: Int. J.* **2010**, *37*, 421–426. [CrossRef]

3. Bakopoulou, K.; Michalos, G.; Mparis, K.; Gkournelos, C.; Dimitropoulos, N.; Makris, S. A Human Robot Collaborative Cell for Automating NDT Inspection Processes. *Procedia CIRP* **2022**, *115*, 214–219. [CrossRef]
4. Sensors | Free Full-Text | Ultrasonic Non-Destructive Testing System of Semi-Enclosed Workpiece with Dual-Robot Testing System. Available online: https://www.mdpi.com/1424-8220/19/15/3359 (accessed on 18 October 2023).
5. Cooper, I.; Nicholson, I.; Yan, D.; Wright, B.; Liaptsis, D.; Mineo, C. Development of a Fast Inspection System for Aerospace Composite Materials—The IntACom Project: 5th International Symposium on NDT in Aerospace. 13 November 2013. Available online: https://strathprints.strath.ac.uk/61717/ (accessed on 5 May 2023).
6. Ma, P.; Xu, C.; Xiao, D. Robotic Ultrasonic Testing Technology for Aero-Engine Blades. *Sensors* **2023**, *23*, 3729. [CrossRef] [PubMed]
7. Aparicio Secanellas, S.; Gauna León, I.; Parrilla, M.; Acebes, M.; Ibáñez, A.; De Matías Jiménez, H.; Martínez-Graullera, Ó.; Álvarez De Pablos, A.; González Hernández, M.; Anaya Velayos, J.J. Methodology for the Generation of High-Quality Ultrasonic Images of Complex Geometry Pieces Using Industrial Robots. *Sensors* **2023**, *23*, 2684. [CrossRef] [PubMed]
8. Zhou, Z.; Zhang, Y.; Tang, K. Sweep Scan Path Planning for Efficient Freeform Surface Inspection on Five-Axis CMM. *Comput. -Aided Des.* **2016**, *77*, 1–17. [CrossRef]
9. Poole, A.; Sutcliffe, M.; Pierce, G.; Gachagan, A. Autonomous, Digital-Twin Free Path Planning and Deployment for Robotic NDT: Introducing LPAS: Locate, Plan, Approach, Scan Using Low Cost Vision Sensors. *Appl. Sci.* **2022**, *12*, 5288. [CrossRef]
10. Zhang, X.; Liu, S.; Xiang, Z. Optimal Inspection Path Planning of Substation Robot in the Complex Substation Environment. In Proceedings of the 2019 Chinese Automation Congress (CAC), Hangzhou, China, 22–24 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 5064–5068.
11. Sun, Q.; Wan, W.; Chen, G.; Feng, X. Path Planning Algorithm under Specific Constraints in Weighted Directed Graph. In Proceedings of the 2016 International Conference on Audio, Language and Image Processing (ICALIP), Shanghai, China, 11–12 July 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 635–640.
12. Wayahdi, M.R.; Ginting, S.H.N.; Syahputra, D. Greedy, A-Star, and Dijkstra's Algorithms in Finding Shortest Path. *Int. J. Adv. Data Inf. Syst.* **2021**, *2*, 45–52. [CrossRef]
13. Holmes, C.; Drinkwater, B.; Wilcox, P. Post-Processing of the Full Matrix of Ultrasonic Transmit-Receive Array Data for Non-Destructive Evaluation. *NDT E Int.* **2005**, *38*, 701–711. [CrossRef]
14. Liaptsis, D.; Yan, D.; Cooper, I.; Papadimitriou, V.; Roditis, G. Development of an Automated Scanner and Phased Array Ultrasonic Testing Technique for the Inspection of Nozzle Welds in the Nuclear Industry. In Proceedings of the 9th International Conference on NDE in Relation to Structural Integrity for Nuclear and Pressurized Components, Seattle, WA, USA, 22–24 May 2012. Available online: https://www.ndt.net/?id=14746 (accessed on 12 March 2023).
15. Khan, A.; Mineo, C.; Dobie, G.; MacLeod, C.; Pierce, G. Vision Guided Robotic Inspection for Parts in Manufacturing and Remanufacturing Industry. *J. Remanufacturing* **2020**, *11*, 49–70. [CrossRef]
16. Corsini, M.; Cignoni, P.; Scopigno, R. Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 914–924. [CrossRef] [PubMed]
17. Chen, J.; Luo, C.; Krishnan, M.; Paulik, M.; Tang, Y. *An Enhanced Dynamic Delaunay Triangulation-Based Path Planning Algorithm for Autonomous Mobile Robot Navigation*; Casasent, D.P., Hall, E.L., Röning, J., Eds.; Electronic imaging: San Jose, CA, USA, 2010; p. 75390P.
18. Foulds, L.R. *Graph Theory Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; ISBN 978-1-4612-0933-1.
19. Cheng, K.P.; Mohan, R.E.; Nhan, N.H.K.; Le, A.V. Graph Theory-Based Approach to Accomplish Complete Coverage Path Planning Tasks for Reconfigurable Robots. *IEEE Access* **2019**, *7*, 94642–94657. [CrossRef]
20. Bento, L.M.S.; Boccardo, D.R.; Machado, R.C.S.; de Sá, V.G.P.; Szwarcfiter, J.L. *Dijkstra Graphs*. CoRR 2016, abs/1602.08653. Available online: http://arxiv.org/abs/1602.08653 (accessed on 4 April 2023).
21. Pinkham, R.; Zeng, S.; Zhang, Z. QuickNN: Memory and Performance Optimization of k-d Tree Based Nearest Neighbor Search for 3D Point Clouds. In Proceedings of the 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), San Diego, CA, USA, 22–26 February 2020; pp. 180–192.
22. Bentley, J.L. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* **1975**, *18*, 509–517. [CrossRef]
23. Hu, L.; Nooshabadi, S. Massive Parallelization of Approximate Nearest Neighbor Search on KD-Tree for High-Dimensional Image Descriptor Matching. *J. Vis. Commun. Image Represent.* **2017**, *44*, 106–115. [CrossRef]
24. Poole, A.; Sutcliffe, M.; Pierce, G.; Gachagan, A. A Novel Complete-Surface-Finding Algorithm for Online Surface Scanning with Limited View Sensors. *Sensors* **2021**, *21*, 7692. [CrossRef] [PubMed]