Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering,
Volume 232, Issue 3, March 2018, pp. 270-284
DOI:10.1177/0959651817751317

Quasi-Real-Time Confined Environment Path Generation for Mobile Robotic Manipulator Arms

D. Galvão Wall[1], J. Economou[2], K. Knowles[3], Cranfield University[4]

Abstract

Path generation for mobile robotic manipulator arms is challenging in dynamic environments because high-speed calculations are required to deal with fast-moving obstacles. A novel path-planning algorithm has been developed which solves in quasi-real time the problem of path generation in confined environments for interconnected multi-body systems, specifically a robotic manipulator arm with three links. This work presented in this paper builds upon the previous work by reformulating the technique to increase the speed at which the algorithm is able to calculate a safe path. The complexity of the task space has increased substantially compared to previous work and the algorithm has been reformulated to speed up the calculation in order to maintain or even improve its ability to plan a safe path in real time. The method is now able to calculate a safe path through environments significantly more quickly than the previous method, and the results presented in this paper expand the complexity of the environment by a large amount and test the ability of the reformulated algorithm to still operate in real time, which the method achieves. It was found that the reformulated method reduces the calculation time for path generation exponentially when used to plan safe paths through test environments involving different numbers of obstacles. The new algorithm thus has the potential to facilitate path planning in challenging dynamic environments, such as those used in sensitive manufacturing and maintenance tasks as well as bomb disposal and similar applications.

Keywords

Robotic manipulator guidance, environment mapping, graph theory, close proximity obstacle avoidance.

Introduction

Robotic arms and their corresponding guidance systems [1] are used for many purposes including manufacturing [2], wheelchair assistance [3], search and rescue [4], [5], surgery [6], nuclear reactor maintenance [7] and improvised explosive device (IED) disposal [8]. The application determines the nature of the control and guidance method, as was previously reported [1]. Appropriate path planning methods include inverse kinematic solutions, [9], [10], incorporating a numerical approach [11], neural networks [12], fuzzy-inference systems [13], vision-based control [14] and potential fields [15]. Other solutions involve rapidly exploring random trees [16], grid-based methods [17] or

[1]Teaching Fellow: d.galvaowall@cranfield.ac.uk
[2]Senior Lecturer
[3]Professor of Aeromechanical Systems
[4]Aeromechanical Systems Group, Centre for Defence Engineering, Cranfield University, Defence Academy of the United Kingdom, Shrivenham, Swindon, SN6 8LA, UK}

genetic algorithms [18]. Although the techniques presented in the current literature are able to provide obstacle avoidance which protects the entirety of a robotic arm from collision with obstacles (an ability which is crucial when operating in restricted operating volumes such as those found in IED disposal or search and rescue applications), they are often unable to solve the problem in three dimensional space or in real time without a priori knowledge of the environment. No current methods satisfy all the criteria specified above and in the previous work addressed these issues [1]. The term “real time” is used extensively in the literature but is subjective to the application and environment. For example, path planning for space-based manipulators working in micro-gravity [19] can be carried out in real time because long calculation times are permissible given the slow movement of the robot arm. Conversely, path planning in dynamic environments [20], where obstacles move more quickly, requires high-speed calculations. This is a challenge because more complex environments require more extensive calculations, therefore increasing the time taken to generate a path. It was previously reported "a novel technique for the control of robotic manipulator arms with many degrees-of-freedom" [1]. This allowed the automatic placement of the end effector in a defined spatial location while ensuring that none of the arm components made contact with obstacles, but was limited to relatively simple and confined environments. Here more complex and realistic scenarios are considered in which a more efficient algorithm is required because the calculation time for the original method increases exponentially with increasing numbers of obstacles [21]. The term 'quasi-real-time' is introduced in this paper because the method described previously and reformulated herein does not achieve an instantaneous result, but generates a path quickly enough for this application and hence can be considered to be a real-time process.

Section 2 of this article outlines the challenge of reducing the number of processes in the path generation method to limit the calculation time. Section 3 describes the parameters of the three-degrees-of-freedom (3-DoF) robotic manipulator arm used to present the technique. Section 4 presents the method used to generate a path for the robotic arm. Section 5 compares the modified method with the original method to confirm the improvements. Finally, the conclusions are presented in section 6.

Formulation of the Problem

The previous method [1] allowed us to plan a path for a 3-DoF robotic manipulator resulting in the successful navigation of a close-proximity environment. The method was created from a series of processes, each of which carried out a single function. Figure 1 is a flowchart showing the relationship between these processes. Block *A* represents the generation of obstacle data, blocks *B* to *M* describe the generation of a map in the configuration space (C-space) of the manipulator arm from measured data in the task space (T-space), which is the XYZ domain of the environment where the obstacle is located (physical domain). Blocks *N*, *O* and *P* represent the path generation through the C-space map. This version of the method contains processes which are repeated multiple times, many of which can be removed if the ordering of the technique is altered.

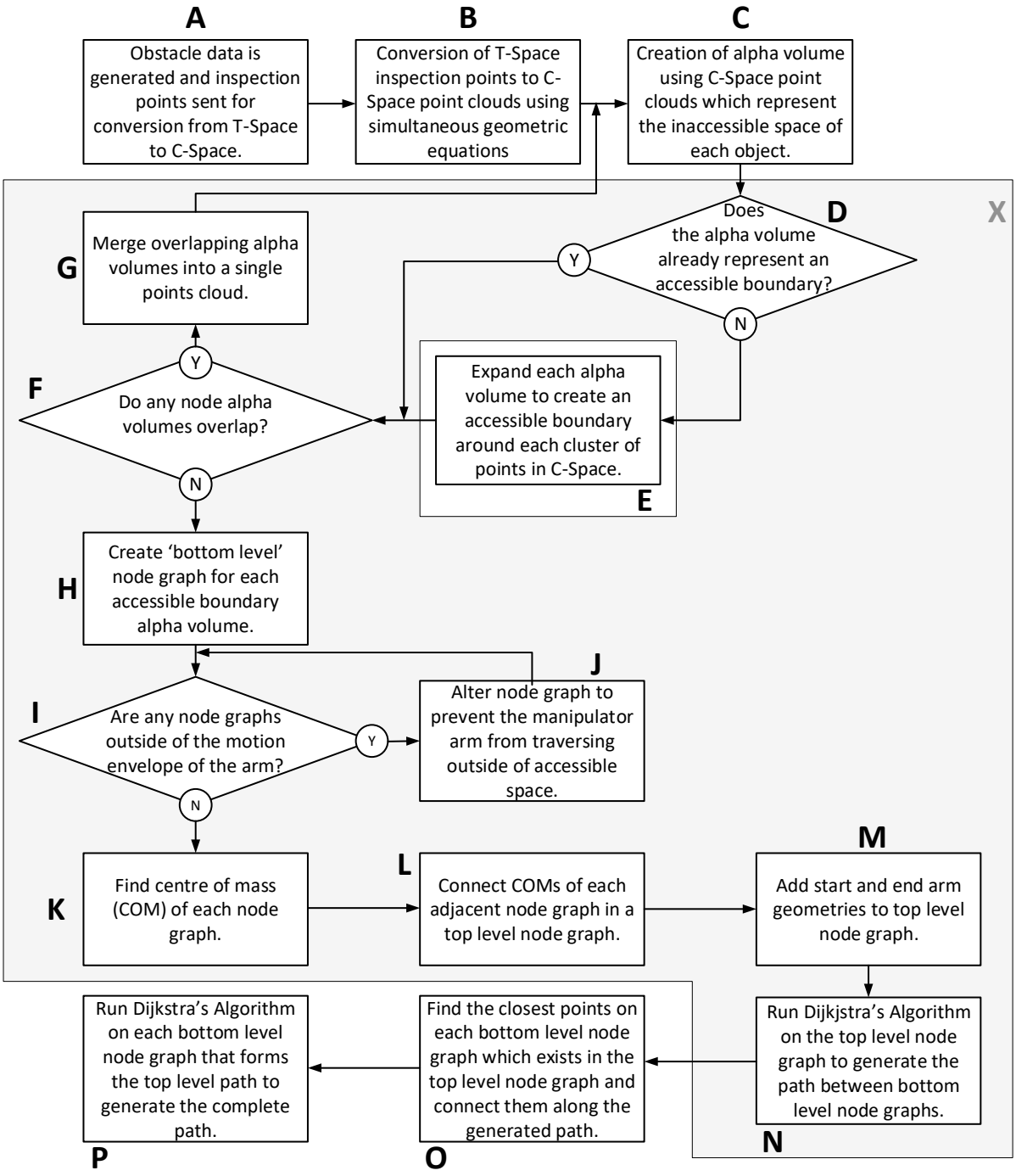

Figure 1: A breakdown of functions that make up the original path generation method [1]. The highlighted area *X* represents the parts of the method which are reformulated in this paper.

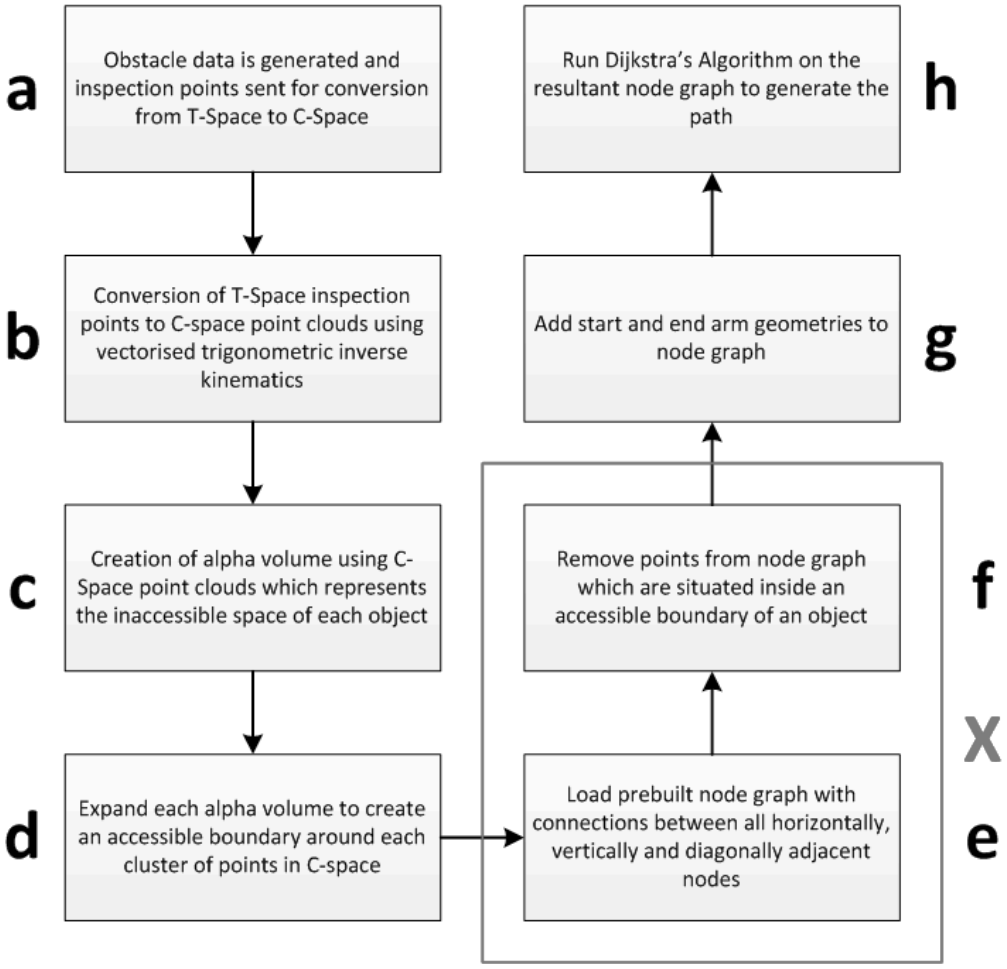

Figure 2: A breakdown of the functions that make up the method presented in this paper. The highlighted processes (*X*) are those which replace *X* in Figure 1.

Robotic Manipulator Arm Parameters

As in the original method [1], a simple model of the 3-DoF robotic manipulator was used to illustrate the technique presented herein. The arm is comprised of three links. The first of the links in the arm is 0.090 m in length and has an axis of rotation about a pivot at the base of the arm. This link rotates in the horizontal (X-Y) plane. The second and third links both operate by rotating about horizontal axis which are perpendicular to the direction of the first link, therefore the plane in which they operate is vertical. The second link has a length of 0.332 m and the third link has a length of 0.538 m.

Figure 3 shows the manipulator arm configuration. L_1, L_2 and L_3 are the lengths of each of the links, xy_1, xy_2 and xy_3 are the xy component of each of the links, and z_2 and z_3 are the z-components of the second and third links, respectively. P_{ef} is the position of the end effector on the third link, and has two components (P_{ef-xy} and P_{ef-z}) that are the x-y and z coordinates of the end effector respectively. P_0 is the position of the first joint and the origin of the arm. The joint angles are represented by α, σ and η. The physical parameters of the robotic manipulator used in this work are outlined in Table 1. The joint angles in this manipulator arm have accessible ranges of $-180° \leq \alpha \leq 180°$, $-40° \leq \sigma \leq 180°$ and $-180° \leq \eta \leq 140°$.

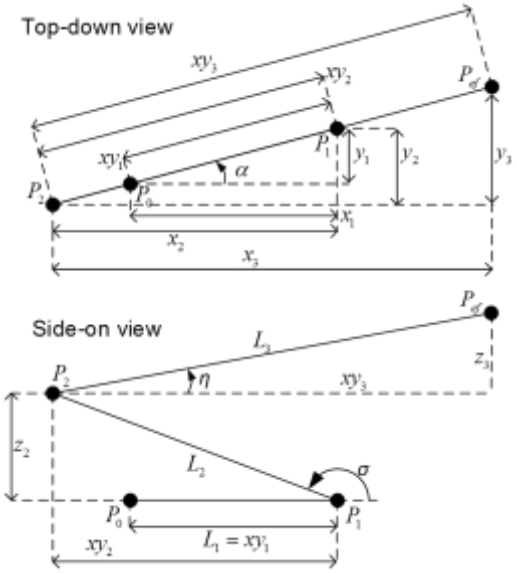

Figure 3: Schematic of the 3-DoF manipulator arm used in this work.

Link Label	Joint	Joint Angle	Link Length
L_1	$P_0 - P_1$	α	$0.090\ m$
L_2	$P_1 - P_2$	σ	$0.332\ m$
L_3	$P_2 - P_{ef}$	η	$0.538\ m$

Table 1: Parameters of links in the 3-DoF manipulator arm.

Path Generation Method

Both the original method [1] and the reformulation presented herein use graph theory [22], [23], [24], [25] in C-space to solve the problem of simultaneous path generation for interconnected multi-body systems in close-proximity environments, but the method presented in this paper includes several modifications. Whereas the original method uses convoluted inverse kinematics to solve the T-space to C-space conversion based on a combination of geometry and simultaneous equations, the new method uses a vectorised trigonometrical solution to achieve the same conversion of object inspection points. The following description mirrors the flow of processes shown in Figure 2.

Obstacle Data Generation

This part of the method is carried out in block *a* of Figure 2. This process simulates the data that could be obtained by a ranging sensor, e.g. based on ultrasonic or laser range finding. This type of sensor detects points in space with a bearing and range, which can be transformed into Cartesian coordinates. For the purposes of this investigation, the generated obstacles take the form of spheres of varying resolution (number of inspection points per sphere), where the inspection points on the surface of the sphere are given as Cartesian coordinates.

T-Space to C-space Conversion

Block *b* in Figure 2 converts the Cartesian inspection points on the surface of each sphere in T-space into C-space. This requires the calculation of the manipulator joint angle combinations (or range of combinations) which cause a collision between the manipulator and the inspection points. For the first and second joint angles, the solution presented herein involves finding the angle between the inspection point and the joint and then comparing the range from the measured point to the joint with the length of the link. If the Cartesian distance between the joint and the measured point is less than or equal to the length of the link then there is a collision with the obstacle at that joint angle.

For joint α, the vectors P_x and P_y are the vectors which contain the X and Y coordinates of each of the measured points in the obstacle, whereas $P_z = 0$ (i.e. the Z coordinates of the measured points lie on the plane of the first link). The α angles of collisions between the first link and any obstacles are calculated using Equation 1, where parameter A is the vector of joint angles found by the inverse tangent of the quotient of the vectors P_y and P_x. For each of these angles, the Cartesian distance to the point from the origin is found using Equation 2 and for all those points where the distance is less than or equal to the length of the first link l_1 the angle is kept as a collision angle.

$$A = \tan^{-1}\left(\frac{P_y}{P_x}\right) \tag{1}$$

$$R = \sqrt{P_x^2 + P_y^2} \tag{3}$$

For joint σ, α must first be calculated using Equations 1 and 2, then the location of the joint between the first and second links can be found using forward kinematics. There are two possible solutions to α, where $-\pi^c \leq \alpha_1 \leq \pi^c$, and where $\alpha_2 = \alpha_1 \pm \pi^c$. The second possible range of α angles is given by Equation 3. The two sets of solutions for σ, and the Cartesian distance from the joint to the inspection point, can be found using the collision geometries shown in Figure 4, and calculated using Equations 4 and 5. If $R < l_2$ then there is a collision. where l_1 and l_2 are the lengths of links 1 and 2 of the manipulator arm, σ_1 and σ_2 are the joint angles of the second link and θ is the calculable angle, which is used to calculate σ_2. P is the Cartesian location of the end of the second link of the arm.

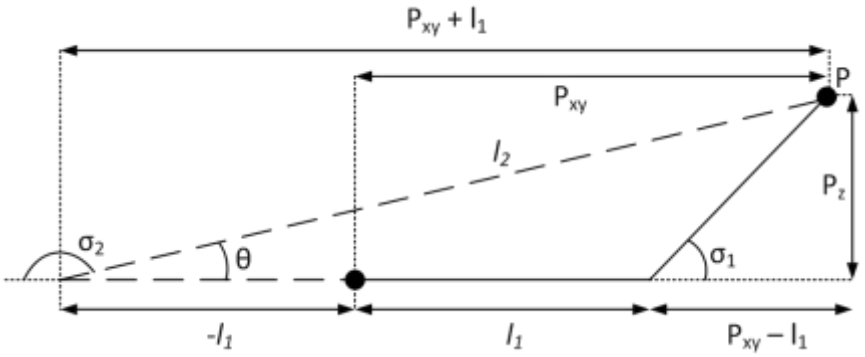

Figure 4: Arm geometry for the calculation of σ.

$$A_1 = A \; and \; A_2 = A \pm \pi^c \tag{3}$$

$$\Sigma_1 = \tan^{-1} \frac{P_z}{\sqrt{P_x^2 + P_y^2} - l_1}$$

$$R_1 = \sqrt{\left(\sqrt{P_x^2 + P_y^2} - l_1\right)^2 + P_2^2}, for \; \sigma_1 \tag{4}$$

$$\Sigma_1 = \tan^{-1} \frac{P_z}{\sqrt{P_x^2 + P_y^2} + l_1}$$

$$R_2 = \sqrt{\left(\sqrt{P_x^2 + P_y^2} + l_1\right)^2 + P_2^2}, for \; \sigma_2 \tag{5}$$

For joint η, Equations 1 and 3 are again used to calculate α. For angles σ and η, the entirety of the third link is taken into consideration because there are multiple solutions for collisions involving the third link and an obstacle. The length of the third link (l_3) is considered to be a vector from 0 m to l_3. The collision geometries for the third link are shown in Figure 5. In this case, O represents the vector coordinates of the base of the manipulator arm, and P represents the vector coordinates of a measured point (or all the measured points in the obstacle).

The geometry in Figure 5 is used to calculate the joint angle combinations for σ and η. The Cartesian distance (r_1 and r_2) between the inspection points and joint σ can be used along with l_2 and l_3 to find all of the internal angles of the triangle made by the three links. The angles θ_1 to θ_6 are internal angles and r_1 and r_2 internal lengths to the geometry used to calculate σ_1 to σ_4 and η_1 to η_4, which are the second and third joint angle solutions to collisions between the third link in the manipulator

and an obstacle. Point B in the figure represents the location of the base of the manipulator, about which the first link rotates.

Equations 6 to 11 can be used to determine the internal angles of the geometries shown in Figure 5. Given the length of the third link as a vector of points along the length of the link, l_3, Equations 8, 9 and 10 provide the angles θ_1 to θ_6 shown in Figure 5. This set of equations provides the necessary parameters to calculate the σ and η values in Equations 10 and 11.

$$r_1 = \sqrt{\left(\sqrt{P_x^2 + P_y^2} - l_1\right)^2 + P_z^2}\ for\ \sigma_1, \sigma_2\ and\ \eta_1, \eta_2$$
$$r_2 = \sqrt{\left(\sqrt{P_x^2 + P_y^2} + l_1\right)^2 + P_z^2}\ for\ \sigma_3, \sigma_4\ and\ \eta_3, \eta_4 \quad (6)$$

$$\theta_1 = \tan^{-1}\frac{P_z}{\sqrt{P_x^2 + P_y^2} - l_1}$$
$$\theta_2 = \tan^{-1}\frac{P_z}{\sqrt{P_x^2 + P_y^2} + l_1} \quad (7)$$

$$\theta_2 = \cos^{-1}\frac{l_2^2 + r_1^2 - l_3^2}{2 l_2 r_1}$$
$$\theta_5 = \cos^{-1}\frac{l_2^2 + r_2^2 - l_3^2}{2 l_2 r_2} \quad (8)$$

$$\theta_2 = \cos^{-1}\frac{l_2^2 + l_3^2 - r_1^2}{2 l_2 l_3}$$
$$\theta_5 = \cos^{-1}\frac{l_2^2 + l_3^2 - r_2^2}{2 l_2 l_3} \quad (9)$$

$$\begin{aligned} \Sigma_1 &= \theta_1 + \theta_2 \\ \Sigma_2 &= \theta_1 - \theta_2 \\ H_1 &= \pi - \theta_3 \\ H_2 &= \theta_3 - \pi \end{aligned} \quad (10)$$

$$\begin{aligned} \Sigma_3 &= \theta_4 + \theta_5 \\ \Sigma_4 &= \theta_4 - \theta_5 \\ H_3 &= \pi - \theta_6 \\ H_4 &= \theta_6 - \pi \end{aligned} \tag{11}$$

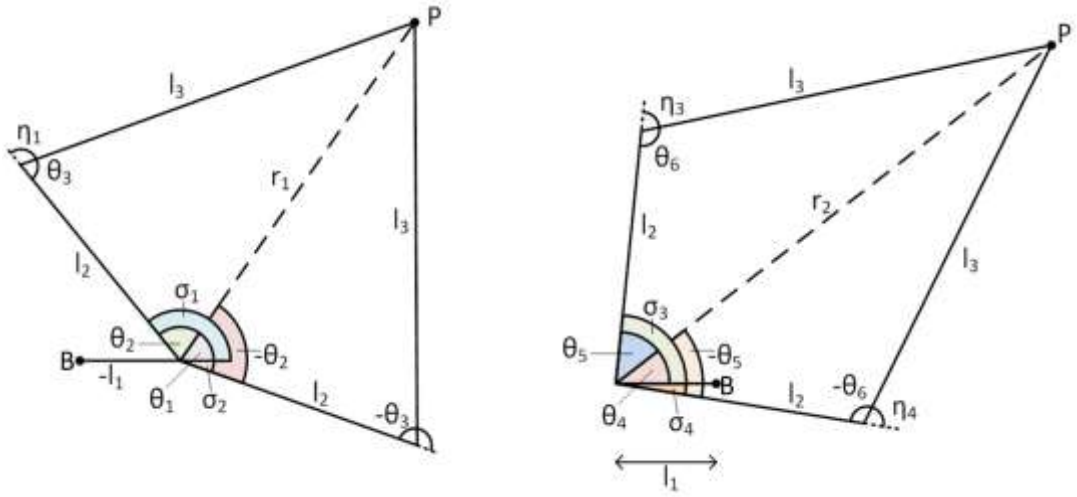

Figure 5: Arm geometry for the calculation of the σ and η joint angle combination.

The calculations described above provide the full set of α, σ and η joint angles for each point in the obstacle for all of the inspection points along the length of the third link. This set of calculations provides the entire range of joint angles that cause a collision between all of the measured points in the obstacle and the entire arm.

Alpha Volume Creation

Block *c* in Figure 2 represents the conversion of the clusters of C-space points into a series of alpha-volumes or concave hulls [26], [21]. These alpha-volumes are Delaunay triangulations that represent the surface of the clusters of points. Figure 6 to Figure 9 show how the alpha-volumes for an obstacle appear in C-space. In figures Figure 6 to Figure 8 the upper plots correspond to obstacle information in T-Space, the black circle represents the manipulator origin and the blue circles represent the cluster of measured points in T-Space. The lower plots represent the C-Space alpha volumes formed by these obstacles. Figure 6a and b shows that as the obstacle is moved further away from the manipulator base (black circle) the c-space inaccessible region gets smaller. This is because as the obstacle moves further away from the manipulator base, the angular range of each joint that would cause a collision is smaller. This can also be observed in Figure 9 (a).

Likewise, the effect of changing the height of the obstacle is presented in Figure a and b. In this case, small changes in obstacle height has only a small effect on the C-space representation of the obstacles. Rather than changing the size of the C-space inaccessible region it has caused a translation of the entire region. This is caused by the shift in obstacle height shifting the range of collision angles slightly rather than widening or narrowing the angle range that causes a collision. Altering the height

much more would cause the range from the manipulator base to the obstacles to change, and the effects described in Figure would have more of an impact.

Figure a and b shows how changing obstacle size affects the shape of the inaccessible region in C-space. It can be seen that smaller obstacles cause smaller inaccessible regions. This can also be observed in Figure 9 (b), which shows why this is the case. A smaller obstacle takes up a smaller angular range in polar coordinates, therefore its size in C-space will be smaller.

Figure 9 (c) shows how changing the obstacle shape will not cause much effect to the inaccessible region in C-space if the angular range that the obstacle takes up is the same.

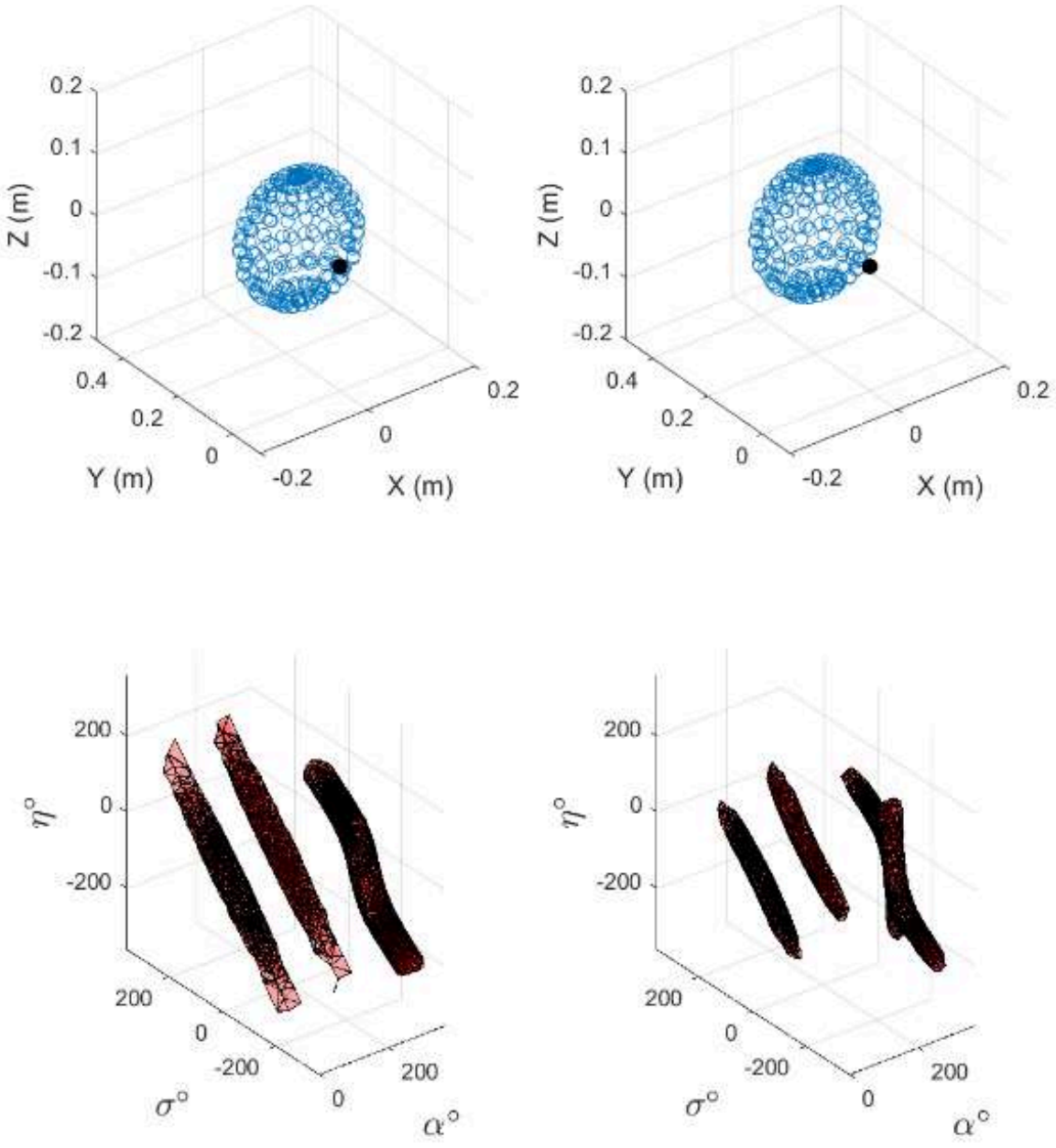

Figure 6a: Change in C-space inaccessible regions with obstacle distance from manipulator base. The plots on the left show an obstacle 0.1 m from the manipulator base and the plots on the right show an obstacle 0.2 m from the base.

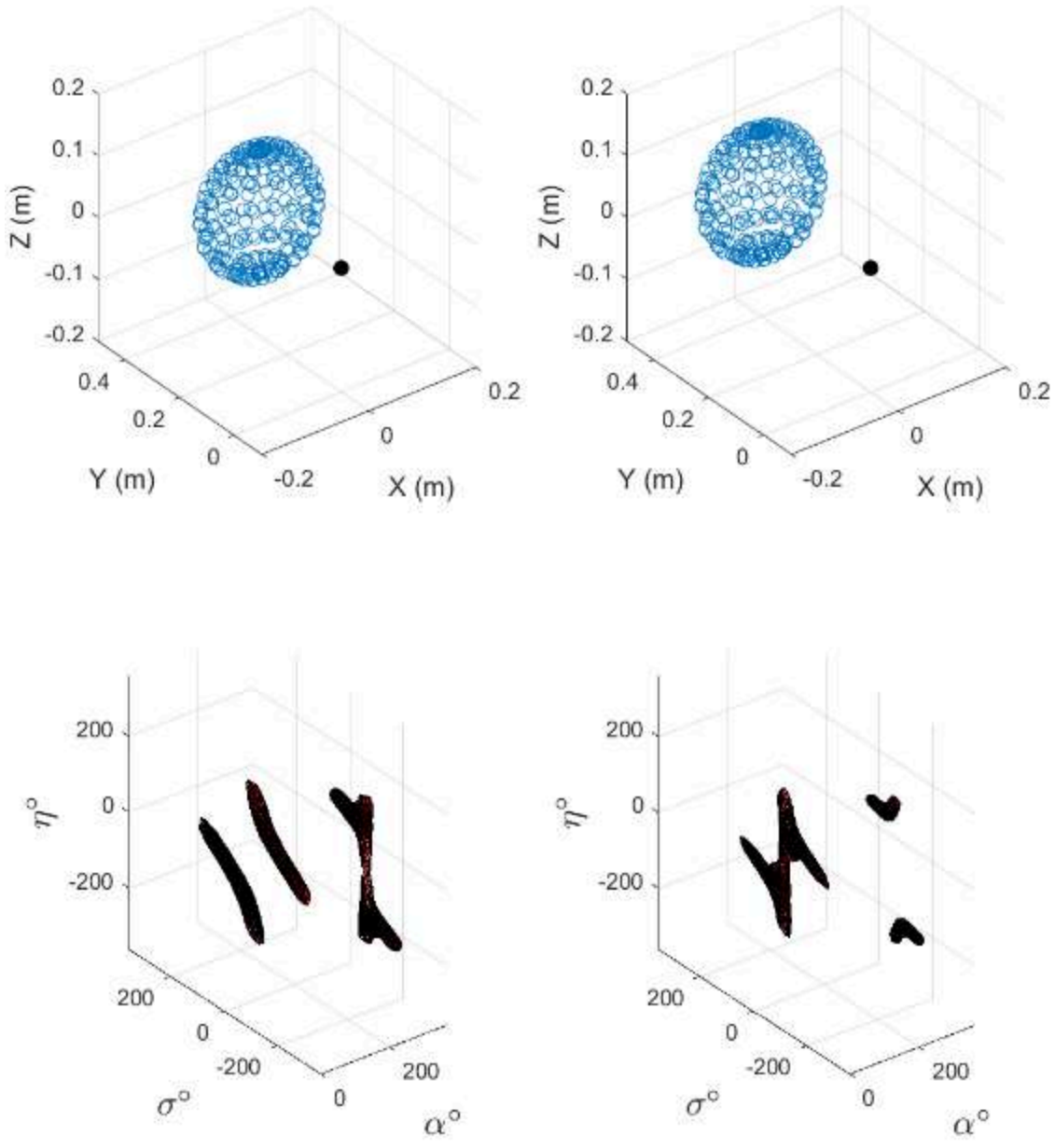

Figure 6b: Change in C-space inaccessible regions with obstacle distance from manipulator base. The plots on the left show an obstacle 0.3 m from the manipulator base and the plots on the right show an obstacle 0.4 m from the base.

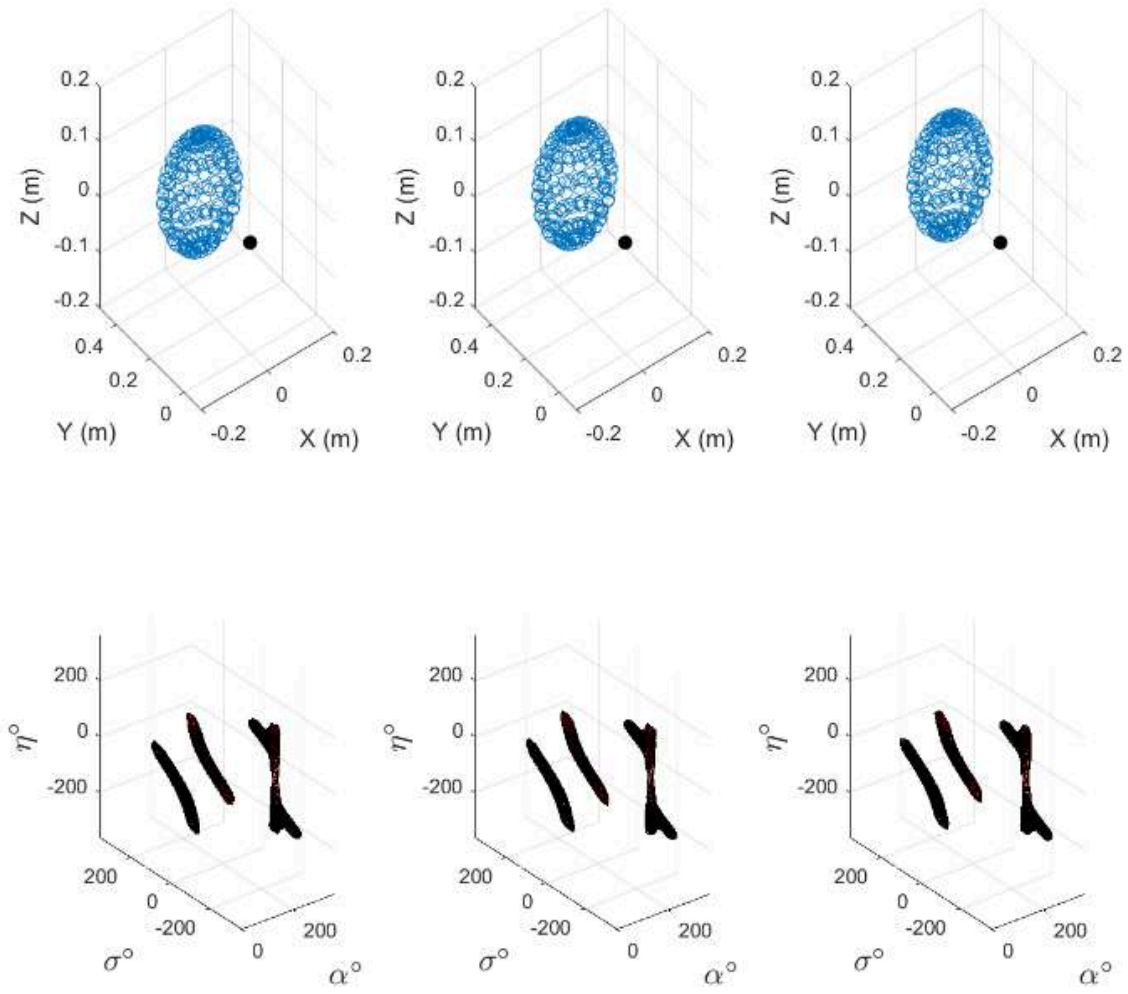

Figure 7a: Change in C-space inaccessible regions with obstacle height. All obstacles have a distance of 0.3 m from the base in the horizontal. The plots on the left show an obstacle with centre at 0.1 m in height, the plots in the middle show an obstacle with centre at 0.05 m in height and the plots on the right show an obstacle with centre at 0 m in height.

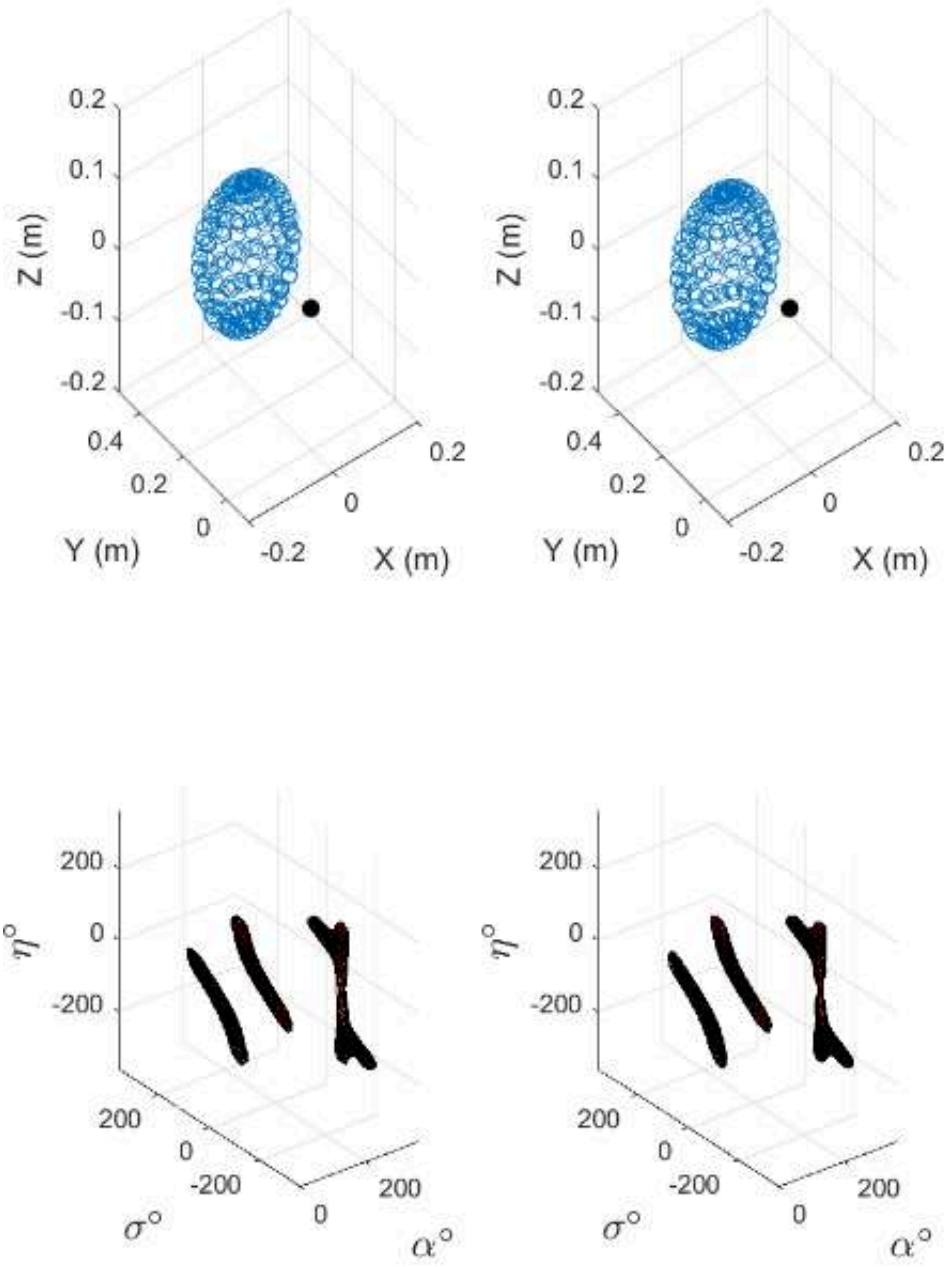

Figure 7b: Change in C-space inaccessible regions with obstacle height. All obstacles have a distance of 0.3 m from the base in the horizontal. The plots on the left show an obstacle with centre at -0.05 m in height and the plots on the right are show an obstacle with centre at -0.1 m in height.

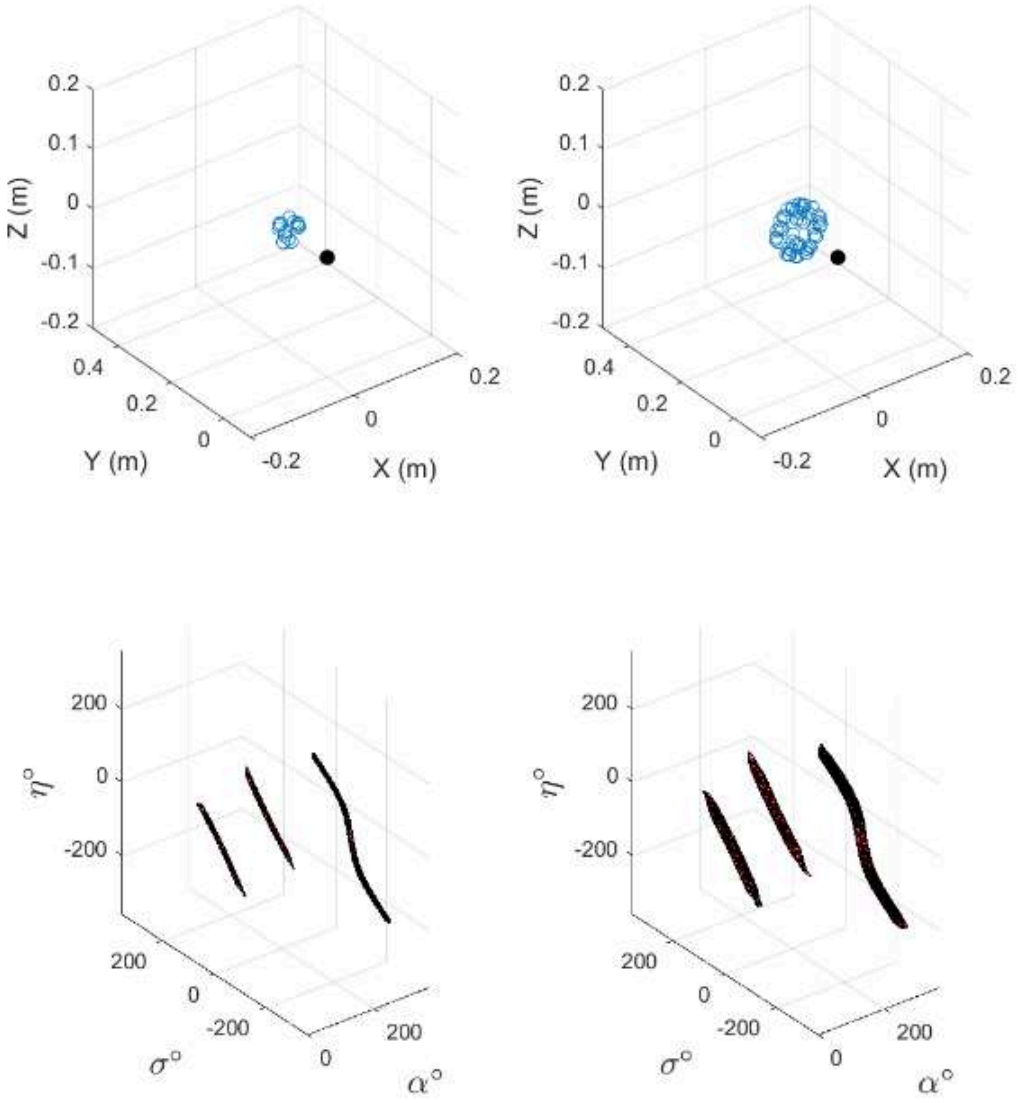

Figure 8a: Change in C-space inaccessible regions with change in obstacle size. All obstacles have a distance of 0.3 m from the base in the horizontal. The plots the left show an obstacle with diameter of 0.05 m. The plots the right show an obstacle with diameter of 0.1 m.

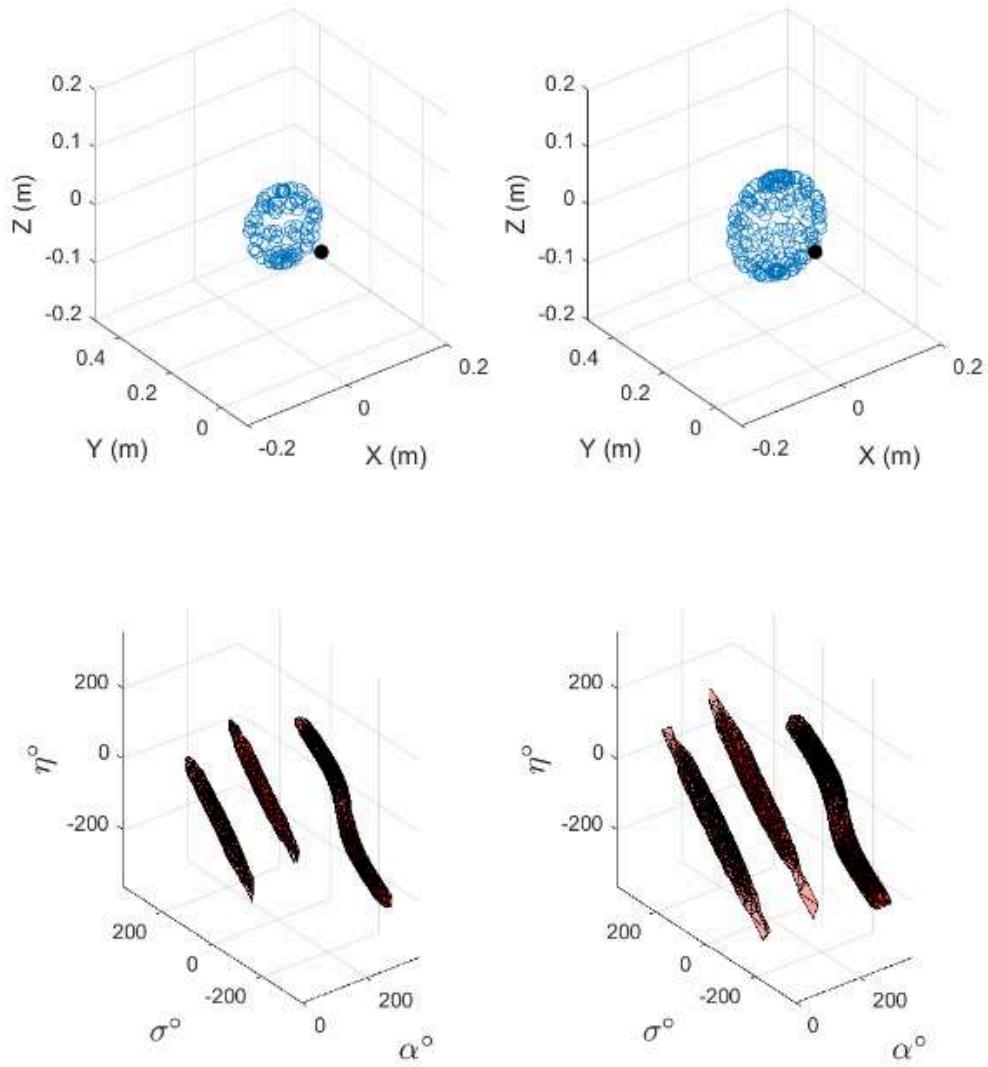

Figure 8b: Change in C-space inaccessible regions with change in obstacle size. The plots the left show an obstacle with diameter of 0.15 m. The plots the right show an obstacle with diameter of 0.2 m.

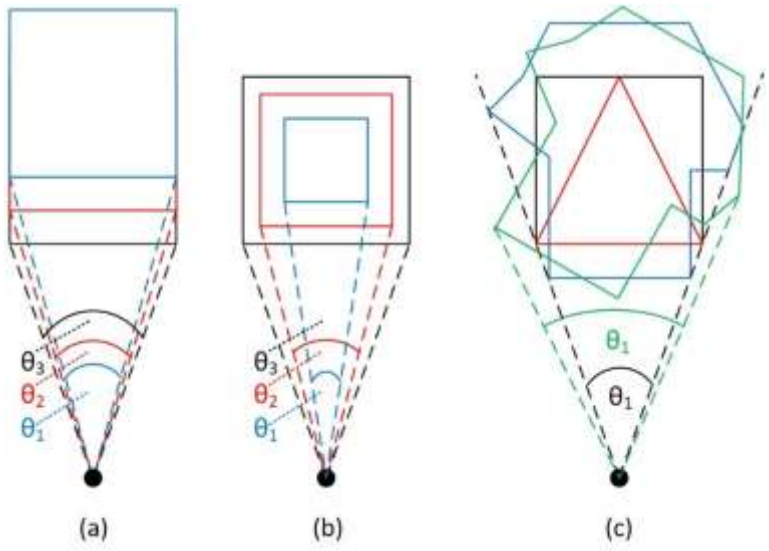

Figure 9: Illustration of how joint angle ranges change with obstacle distance, size and shape.

Expansion of Impermissible Region to Permissible Boundary

For every obstacle in the T-space, there exists a region in C-space to which access by the arm is not permissible. These inaccessible regions can be visualised in a three-dimensional map using the X, Y and Z dimensions as the α, σ and η joint angle ranges of the arm. Moving the arm in such a way that its joint angle combination touches or lies inside one of these impermissible regions would cause a collision with an obstacle in T-space.

To prevent any collisions with the impermissible region formed by the C-space obstacle, the impermissible region can be expanded to form a new shape which is slightly larger than the original C-space shape. This is carried out in block *d* of Figure 2. Expanding the impermissible region creates a new region with an unobstructed boundary. It is possible to navigate along this boundary but the arm must not cross into space which contains an obstacle collision region.

To enact this boundary expansion, each C-space point which forms the alpha-volume becomes the centre of a new sphere of points with a specified radius. A new cluster of points is added for each C-space point on the alpha-volume and then a new alpha-volume is calculated.

There are practical limitations to systems which determine the distance between the permissible boundary of an object and the forbidden region. The first limitation is the potential for error in the lidar sensor (e_s), which could result in the inaccurate measurement of the obstructions. The second limitation is the resolution of the sensor (r_θ) which will cause the sensor to miss the very edges of obstacles when scanning for points in range. A vector was designed to create a series of points around each point in the C-space forbidden region set as shown in Equation 12, where C_{points} represents the vector of points in C-space, $C_{points2}$ represents the new vector of points in C-space, e_l is the angular error in the lidar sensor, r_θ is the angular resolution of the obstacle location measurement, and e_s is the measurement error of servo encoders. This new vector provides a cluster of points that can be used to create a node graph.

$$C_{points2} = C_{points} + (\theta_{1ss} + \theta_l + r_\theta + \theta_2)\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0.7071 & 0.7071 & 0 \\ 0.7071 & -0.7071 & 0 \\ -0.7071 & 0.7071 & 0 \\ -0.7071 & -0.7071 & 0 \\ 0.5774 & 0.5774 & 0.5774 \\ 0.5774 & -0.5774 & 0.5774 \\ -0.5774 & 0.5774 & 0.5774 \\ -0.5774 & -0.5774 & 0.5774 \\ 0.5774 & 0.5774 & -0.5774 \\ 0.5774 & -0.5774 & -0.5774 \\ -0.5774 & 0.5774 & -0.5774 \\ -0.5774 & -0.5774 & -0.5774 \end{bmatrix} \quad (12)$$

Figure 10 shows how the impermissible regions are expanded to create a traversable boundary which allows for safe passage around them. The red volumes represent the impermissible regions formed by the T-space to C-space conversion and the transparent blue volumes represent the permissible boundary around them which have been formed by expanding the impermissible regions by the amount calculated by Equation 12. The boundary expansion has been exaggerated in the figure to highlight the effect of the process.

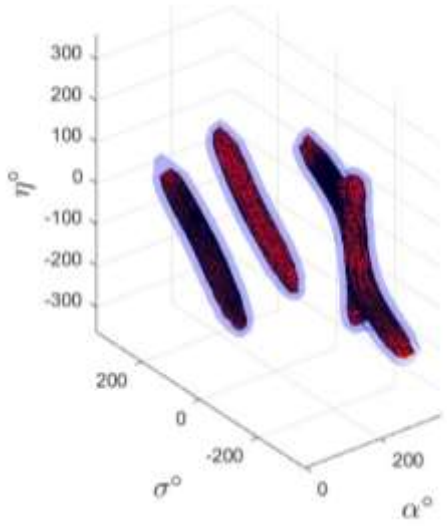

Figure 10: Expansion of the impermissible region in C-space to create a traversable boundary around the obstacles.

Node Graph Formation

In Figure 2, blocks *e* and *f* represent node graph formation. In the earlier method [1], the node graph in C-space was formed using the alpha-volumes of each obstacle. Several algorithms were used to (i)

check for overlaps between permissible boundaries and combine any overlapping boundaries into one; (ii) search permissible boundaries for the closest points between them to connect them together; (iii) search the edges of the space for paths which exceed the limitations of the space and check permissible boundaries in case they exceed the limitations of the space; and (iv) create two different levels of node graph, a global node graph and local node graphs for each permissible boundary to prevent memory problems.

In the refined method, all of these separate steps are replaced with a single process, thus saving computational overheads and calculation time. To achieve this reduction in complexity, the current method requires only two processes. The first creates a node graph which covers the entirety of the space as though there are no obstacles. The second checks whether any of the nodes in this graph fall inside the permissible boundaries of the obstacles in C-space, and if so removes them both from the list of indices and from the adjacency matrix for the space.

The main advantage of this method is that the node graph of the unobstructed space can be generated a priori because it represents the reachable range of the manipulator arm and is therefore time invariant. This empty node graph can be loaded when necessary, saving large amounts of memory. Furthermore, the number of nodes in this graph will only reduce because nodes are removed when they are found to be obstructed. Figure 11 to Figure 15 illustrate the implementation of this method.

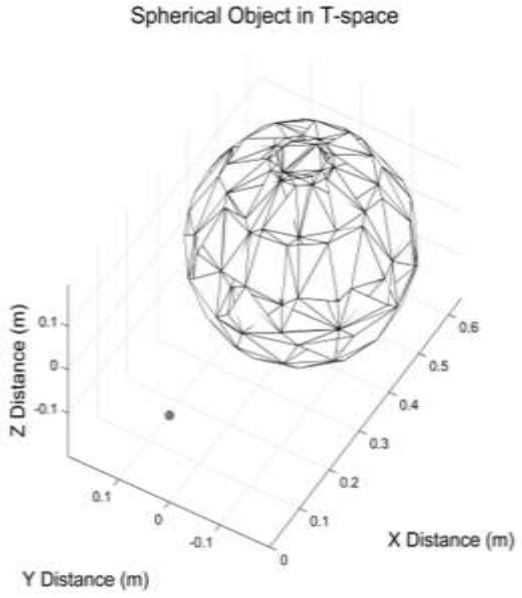

Figure 11: A sphere in T-space in relation to the base point of the manipulator arm.

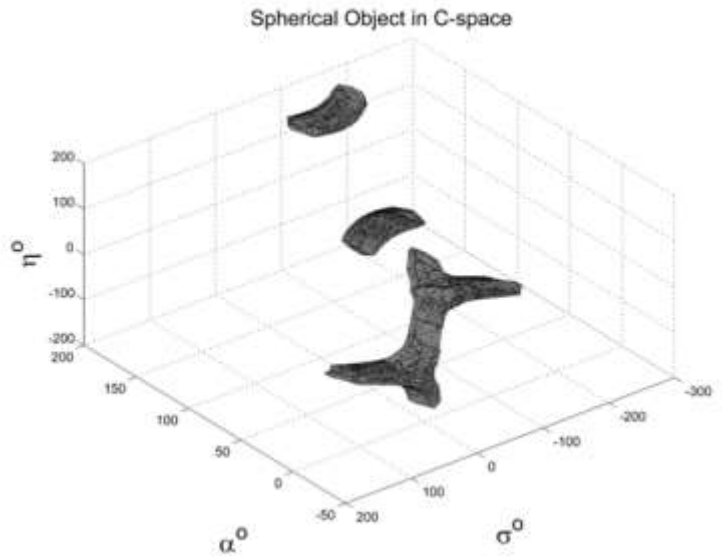

Figure 12: C-space representation of the sphere in relation to the manipulator arm.

The choice of spacing between nodes is important and reflects two limitations, one relating to the control limitations of the manipulator, and the other to the memory capabilities of the processor carrying out the node graph construction.

The first limitation is the steady-state error of the joint angles. Although this is accommodated when the impermissible regions are expanded to form permissible boundaries around which the arm can safely travel, the nodes in the corresponding graph are direct control requirements. Therefore, if they have a smaller separation than the steady-state error of the system, the arm will match the location of two different nodes in reality and also in the guidance method. For this reason, the spacing between nodes must be equal to or larger than the steady-state error of the joints.

The second limitation is that the computer generating the node graph is restricted in terms of memory, and node graphs with a small spacing will contain a large number of nodes, increasing the size of the adjacency matrix in the computer memory, hence increasing overheads and run time. For example, based on the angle range limitations of the manipulator arm used in the experiments, the size of the adjacency matrices for angle spacings of $1°$, $2°$, $5°$ and $10°$ are shown in Table 2.

Node Spacing	$1°$	$2°$	$5°$	$10°$
Adjacency Matrix Size	6.365×10^9	4.036×10^8	1.079×10^7	7.24×10^5

Table 2: Adjacency matrix sizes for node graphs with different degrees of node spacing.

The smaller the spacing between nodes the larger the adjacency matrix of the graph, and this relationship is exponential. Therefore node spacings of $5°$ or 0.087^c were used to reduce the computational overheads. The resulting set of C-space permissible boundaries can be superimposed on the node graph of the empty space. As shown in Figure 13, some of the angle combinations that

cause collisions fall outside this space. However, Figure 14 highlights the nodes in the empty node graph that collide with the forbidden region of the impermissible C-space for the obstacle. The triangulation in Figure 15 shows the space left behind following the removal of the red nodes. The triangulation itself is not used but is shown for illustrative purposes only.

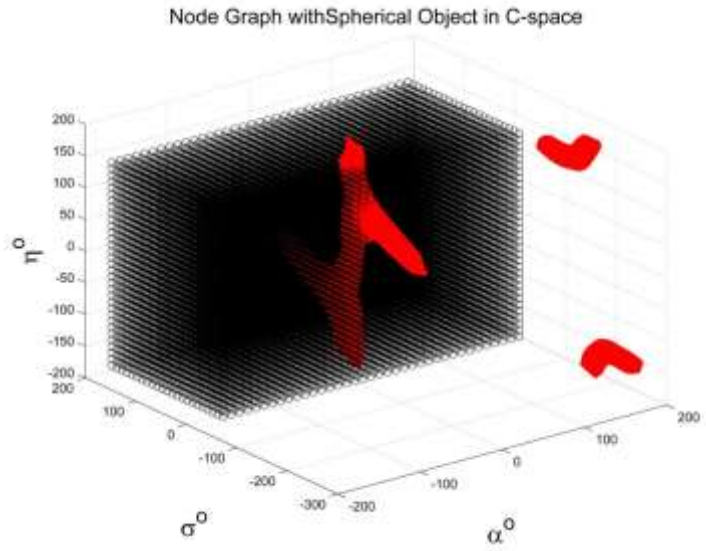

Figure 13: Sphere in C-space (red) superimposed on the unmodified node graph (black).

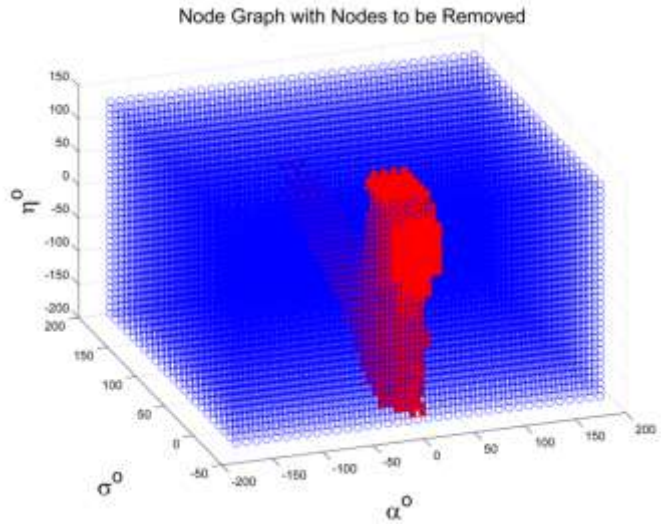

Figure 14: Nodes that collide with the sphere in C-space (red) can be removed from the remainder of the node graph (blue).

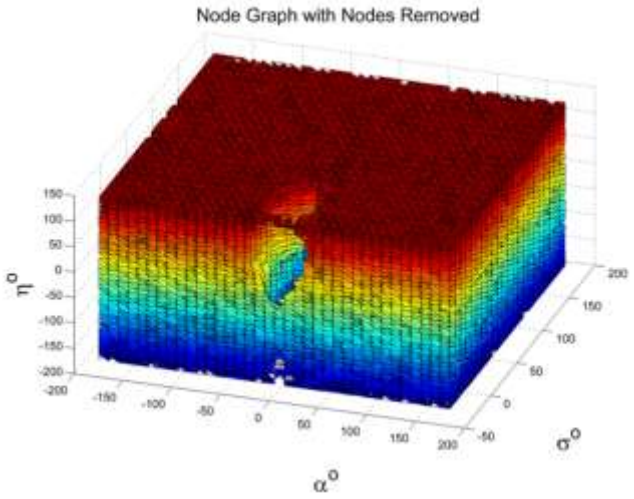

Figure 15: Nodes that collide with the sphere in C-space (red) can be removed from the remainder of the node graph (blue).

The nodes in the graph of the empty space are compared with the permissible boundary regions and any nodes that fall inside these regions can be removed. The red nodes in Figure 14 illustrate which nodes will be removed from the graph.

Mapping the End Effector Start Point and Required Joint Positions

Block *g* in Figure 2 is responsible for adding the end effector start point and required joint positions to the node graph. As for the other inspection points, there are several potential solutions when these points are mapped into the angle space. The solutions with the smallest angular change (i.e. closest together in the space) are selected as long as they do not violate one of the permissible boundaries, making the solution inaccessible. If all solutions are inaccessible then the start and/or required end points are also inaccessible in Cartesian space. The end effector start and required end position are mapped in Figure 16. The green circle represents the angle combination at the start position and the red circle represents the angle combination at the required end position. Table 3 shows the transition from the Cartesian domain to the control domain.

The end effector start and required angle combinations are compared with the centre point of each permissible boundary in the space to find the closest nodes in the graph. These points are then connected solely to the closest node, completing the map of the environment including the end effector start and required end points.

End Effector Point Identifier	T-Space	C-Space
Start Point (P_s)	$\begin{bmatrix} -0.0514\ m \\ -0.2915\ m \\ 0\ m \end{bmatrix}$	$\begin{bmatrix} 100° \\ 0° \\ 0° \end{bmatrix}$
Required End Point (P_r)	$\begin{bmatrix} -0.0514\ m \\ 0.2915\ m \\ 0\ m \end{bmatrix}$	$\begin{bmatrix} 100° \\ 0° \\ 0° \end{bmatrix}$

Table 3: Conversion between T-Space and C-space for the end effector initial and required positions shown in Figure 12.

Generation of a Safe Path using Graph Theory

Block *h* in Figure 2 calculates a safe path through the C-space using Dijkstra's algorithm [22] as previously described [1]. Table 4 lists the joint angle combinations in the path that are required to navigate the end effector of the manipulator arm safely around the spherical obstacle presented in Figure 12. Figure 16 shows the path generated in C-space. The blue point represents the starting location in C-space and the green point the end location in C-space. The red line is the generated path about the obstacle. The area highlighted by the orange has been enlarged to show the path in more detail.

Point ID	α°	σ°	η°
1	-100	0	0
2	-121.695	-14.1908	0.1828
3	-159.75	-42.4559	-49.5
4	165.3354	-27.9574	-49.5
5	144.0931	-10.6416	-49.5
6	125.7179	-4.6067	-11.838
7	100	0	0

Table 4: Joint angle combinations to guide the robotic manipulator from a starting position to a required position while avoiding obstacles.

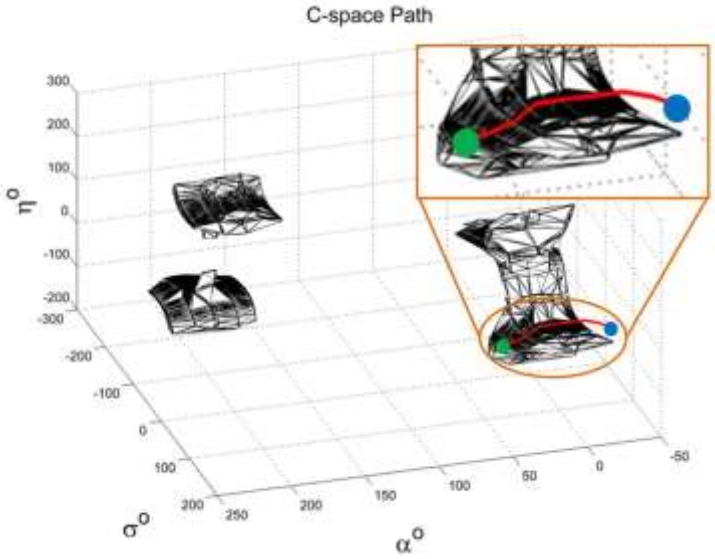

Figure 16: Spherical object in C-space with a safe path generated around it.

Analysis

To illustrate the effectiveness of the new method compared to its predecessor [1], a series of simulations was carried out to assess the difference between the methods in environments of increasing complexity. Uniform spheres of identical size were used to quantify the complexity of the environment. The spheres were always placed at a fixed Cartesian distance from the fixed base of the manipulator arm and two parameters were used to vary the complexity of the environment. The

number of spheres in the environment was varied from 1 to 10, and the number of inspection points per obstacle was varied from $\sim 10^1$ to $\sim 10^6$ by controlling the manner in which the inspection points were generated.

$$\begin{bmatrix} P_X \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} C_x + L\cos\theta_1 \cos\theta_2 \\ C_y + L\sin\theta_1 \cos\theta_2 \\ C_z + L\sin\theta_2 \end{bmatrix} \tag{13}$$

If θ_1 and θ_2 have the same range of increments, i.e. $-180°{:}\,\delta\theta{:}\,180°$, then $\delta\theta$ loosely represents the resolution of the inspection points where $n_i \propto \frac{1}{\delta\theta^2}$. Figure 17 shows the layout of the obstacles in the environment as the number of obstacles is increased.

The following decisions were made about the configuration of the environment: the node spacing in a node graph is fixed, each obstacle is identical in shape and size, the distance between obstacles is equal, and the obstacles are located at an equal distance from manipulator base. The results discussed below represent paths generated around 3 or 10 obstacles.

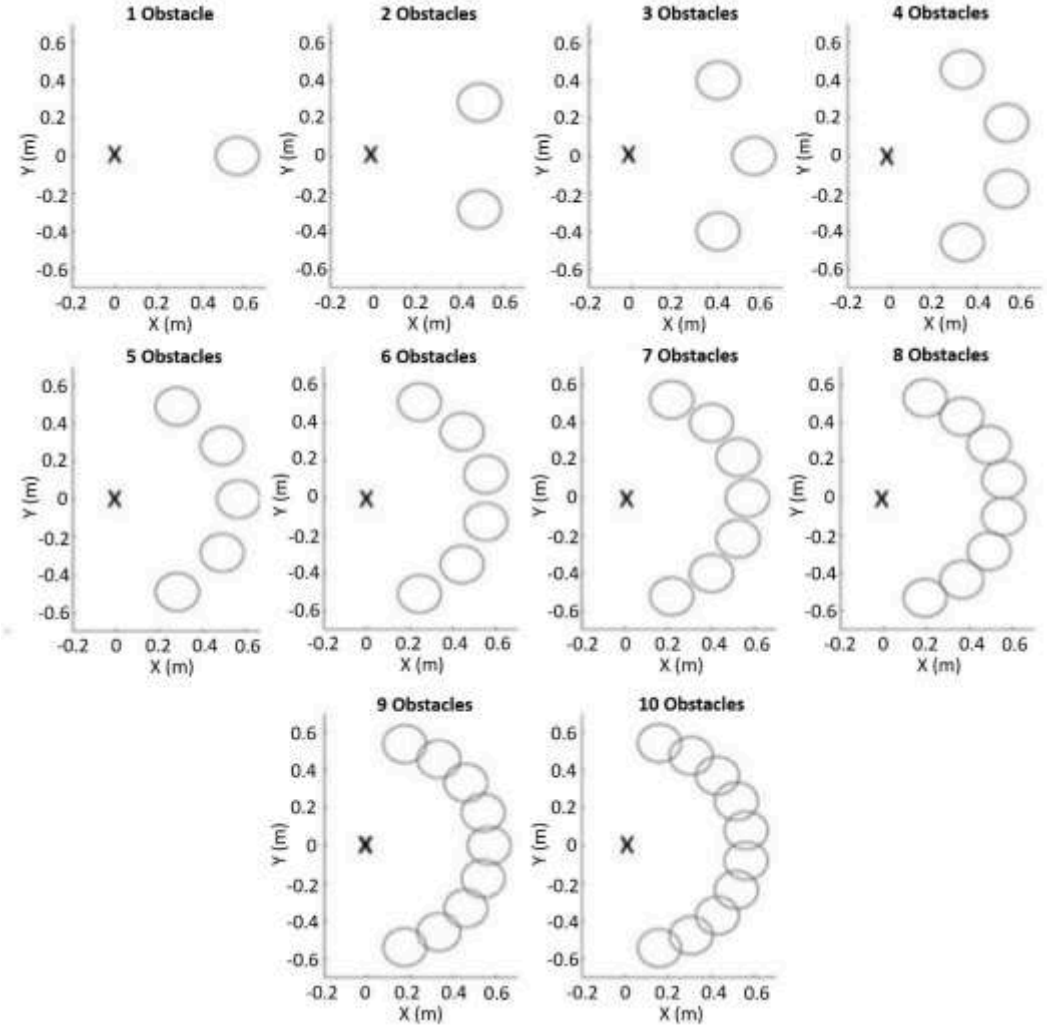

Figure 17: Schematic of the environments used to compare the algorithms.

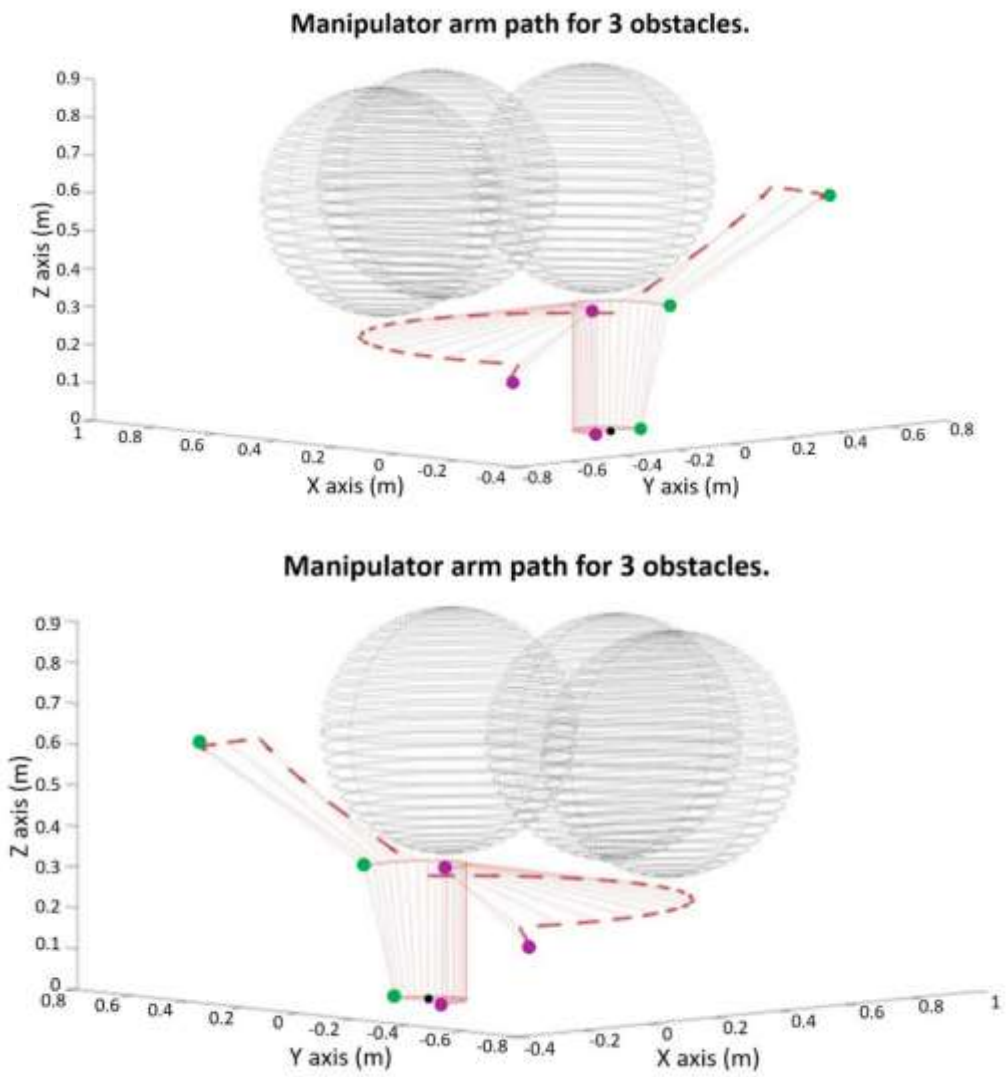

Figure 18: Safe path generated for the manipulator arm around three obstacles.

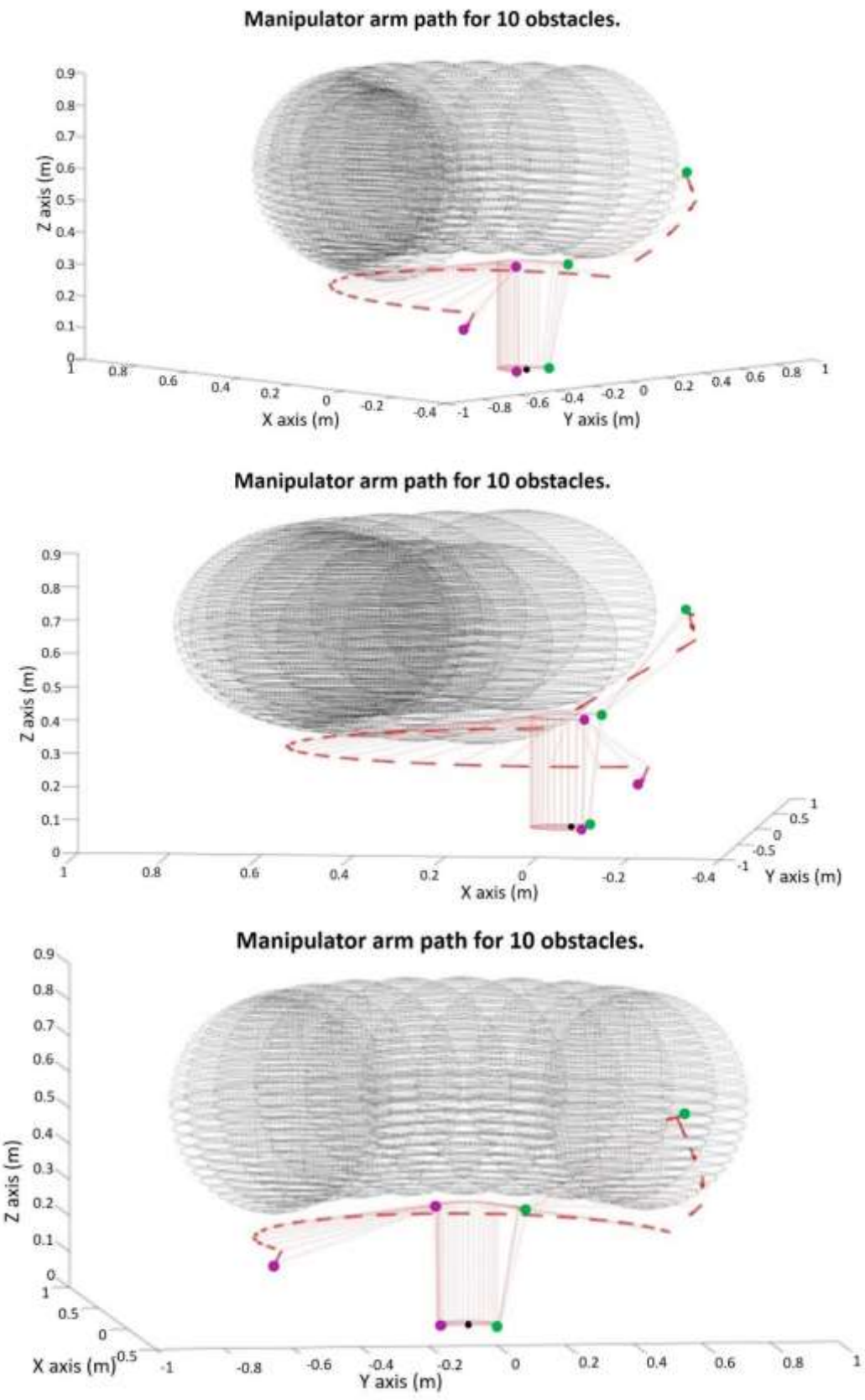

Figure 19: Safe path generated for the manipulator arm around 10 obstacles.

As shown in Figure 18 and Figure 19, the algorithm can plot a safe path in close proximity to the obstacles in both cases. The orientations were chosen to illustrate the clear space between the path and the obstacles, confirming that there is never a collision between the manipulator arm and the environment. The green points indicate the starting positions of the end effector and intermediate joints in the arm, and the magenta points indicate the end positions.

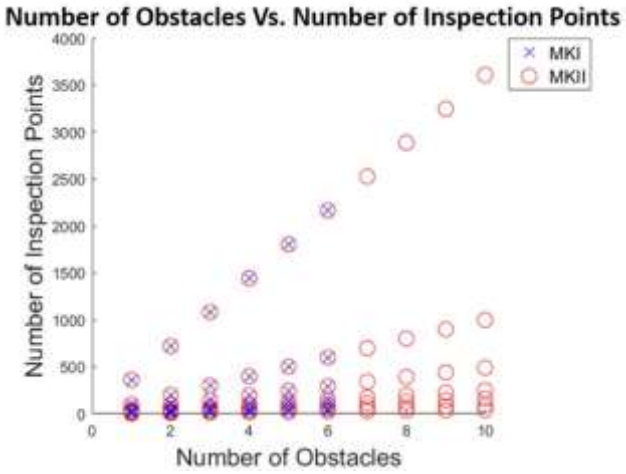

Figure 20: Number of inspection points in the environment when the number of obstacles is increased.

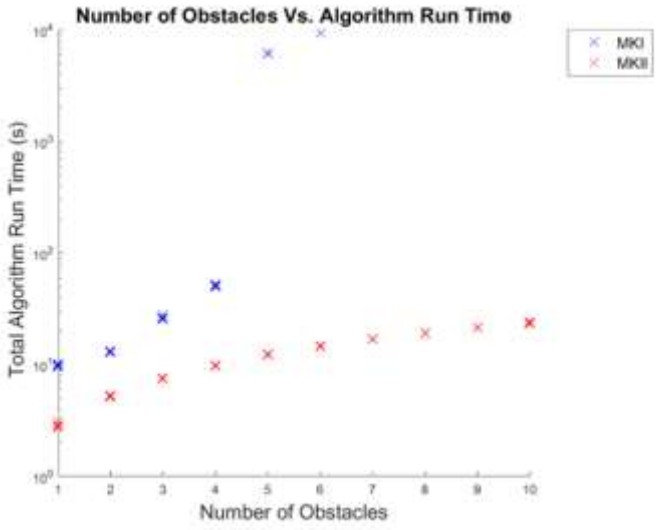

Figure 21: Algorithm run time with increasing numbers of obstacles in the environment.

Figure 20 shows the number of inspection points in the environment plotted against the number of obstacles. This demonstrates the linear relationship between the number of obstacles in the environment and the corresponding number of inspection points. Each gradient represents constant numbers of inspection points per obstacle. There are two clear relationships: first, the data can be organised into sets of vertical lines that represent the same obstacle numbers but with increasing

numbers of inspection points per obstacle as the resolution of the measurements is increased; and second, the series of diagonal lines of increasing gradient starting at the origin represents the number of inspection points increasing with increasing obstacle numbers for a set of constant resolutions. It is important to note that there are no results presented for seven or more obstacles when using the original algorithm because the total calculation time increases exponentially with the number of obstacles, and each calculation takes days or longer when there are seven or more obstacles (Figure 21).

In contrast, the new algorithm has a shorter run time in all cases, and the rate of increase with respect to the number of obstacles is reduced. Figure 20 shows that the relationship between number of obstacles and the run time of the original algorithm [1] (blue crosses) is exponential whereas the same relationship for the algorithm presented herein (red crosses) is logarithmic, suggesting that the calculation time for the original algorithm gets longer with more obstacles, whereas the calculation time for the new algorithm stabilises as the obstacle count increases.

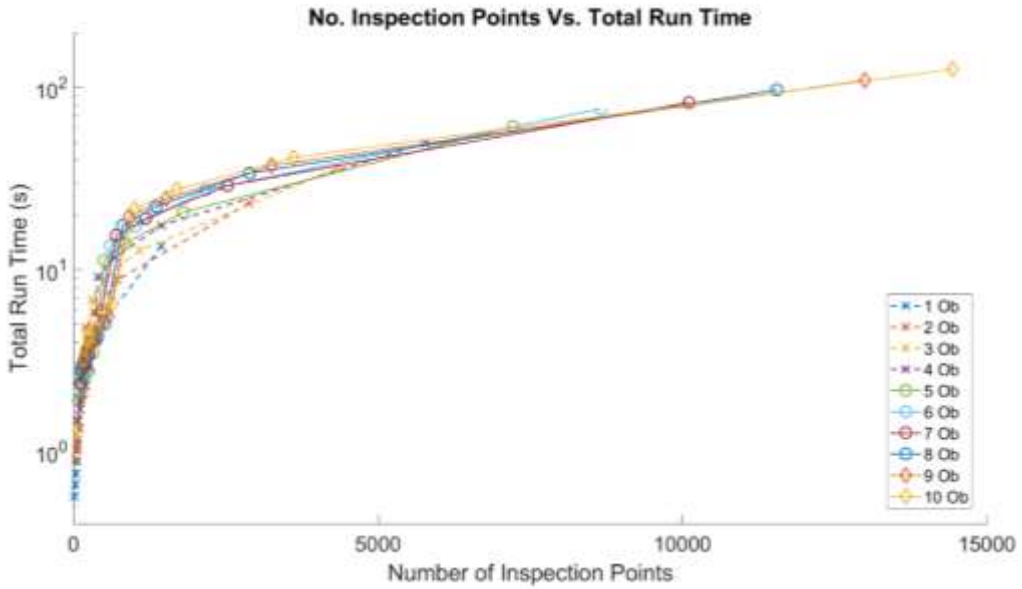

Figure 22: Total algorithm run time with increasing numbers of inspection points.

Figure 22 and Table 5 show how the total algorithm run time is affected by the number of inspection points per obstacle. The dotted lines represent constant numbers of obstacles as the number of inspection points per obstacle is increased. It can be seen by the figure that as the number of inspection points is increased, the total runtime of the algorithm increases. It can also be seen that the algorithm run time increases by a smaller amount as the number of inspection points is increased. Furthermore, the number of obstacles has less effect on the run time as the number of inspection points per obstacle is increased. This is further explained by Figure 23.

		Algorithm Runtime (s) for each No. Inspection Points per Obstacle								
		16	25	36	49	64	100	169	361	1444
No Obstacles	1	0.563	0.656	0.750	0.875	1.094	2.500	3.000	4.484	13.484
	2	0.906	1.016	1.156	1.328	1.938	4.797	5.766	8.781	23.219
	3	1.234	1.453	1.703	2.063	2.109	6.812	8.859	12.766	36.109
	4	1.484	1.859	2.047	2.297	2.859	9.125	11.672	17/484	49.453
	5	1.906	2.406	2.578	2.766	3.469	11.250	13.875	20.656	61.234
	6	2.188	2.594	3.250	3.969	4.094	13.484	17.297	27.344	76.609
	7	2.344	2.938	3.406	4.078	5.891	15.406	19.094	28.969	82.953
	8	2.813	3.172	4.016	4.313	5.031	17.422	22.172	33.875	97.359
	9	3.094	3.641	4.031	4.828	5.531	19.469	24.469	37.234	109.953
	10	3.219	4.266	4.781	6.063	6.531	21.547	27.531	41.359	126.500

Table 5: Numerical Results of Data Presented in Figure 22.

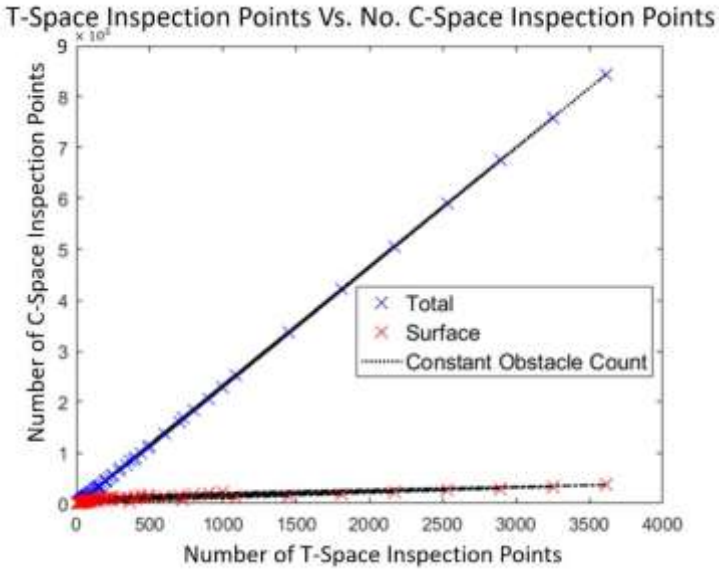

Figure 23: Comparison of the number of inspection points in T-space and in C-space.

As shown in Figure 23, the total number of C-space inspection points increases linearly with increasing numbers of T-space inspection points (blue crosses). The number of C-space inspection points that form the surface of the obstacles in C-space is much smaller and also not linear, resulting in a logarithmic relationship similar to that shown in Figure 20. The connection is that once the T-space to C-space conversion is complete, the remainder of the algorithm operates using only the C-space surface inspection points, and all other C-space points are discarded.

Increasing the number of T-space inspection points in an obstacle increases the total number of C-space inspection points linearly, but more and more of them lie inside the C-space obstacle volume and not on the surface of the volume, hence they are discarded.

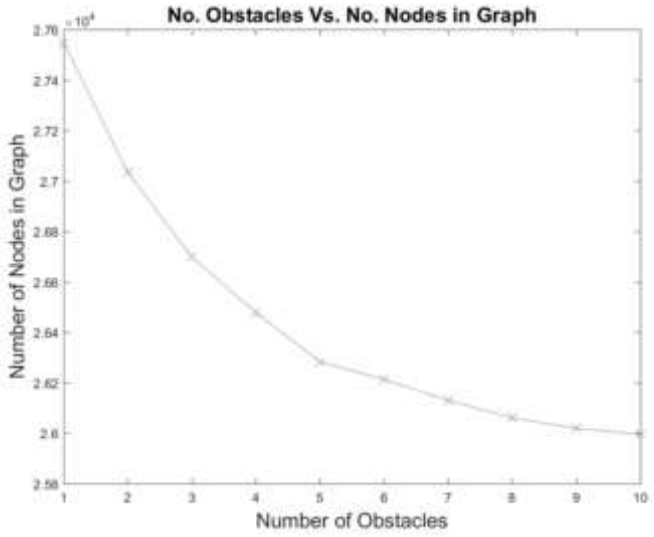

Figure 24: Comparison of the number of nodes in the node graph with increasing numbers of obstacles in the environment.

The way in which the new algorithm deals with node graph generation means that the size of the node graph adjacency matrix reduces with increasing numbers of C-space obstacles. This is because the method preloads a complete node graph with premade adjacency connections representing the complete reach of the manipulator arm in C-space. Nodes are then removed from the graph if they intersect with or fall inside the surfaces formed by the C-space inspection points. This means that for larger numbers of obstacles, there are fewer nodes in the graph and the Dijkstra's algorithm has fewer computations and thus takes less time to calculate a path. This can be observed from the relationship between number of obstacles and the number of nodes in the graph in Figure 24 which shows that the number of nodes reduces with increasing numbers of obstacles.

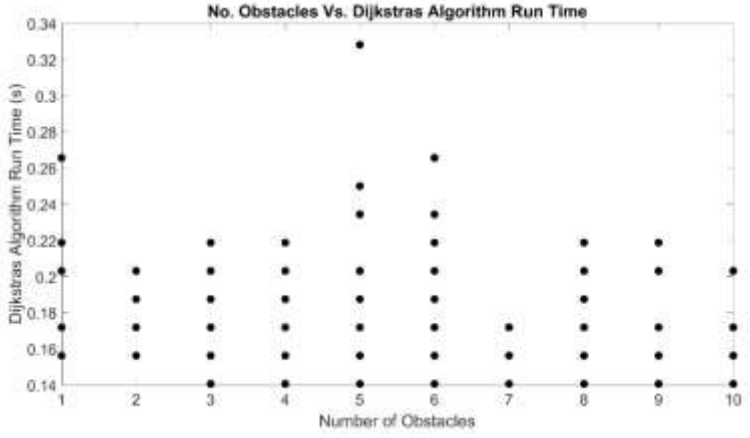

Figure 25: Run time of Dijkstra's algorithm when the number of obstacles in the environment is increased.

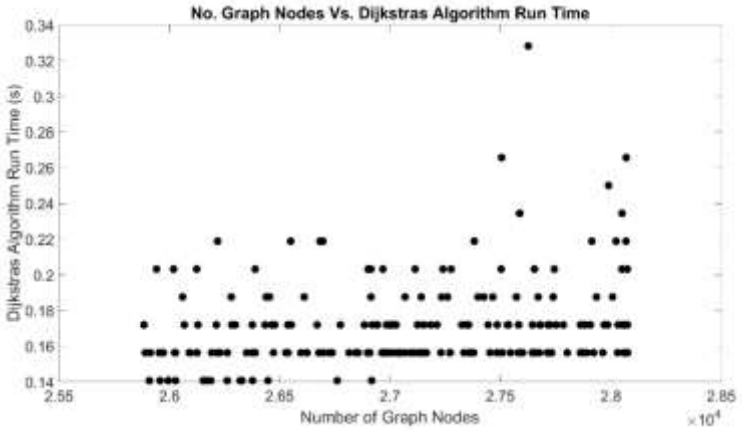

Figure 26: Run time of Dijkstra's algorithm when the number of nodes in the node graph is increased.

The run time of Dijkstra's algorithm is compared to the number of obstacles in Figure 25 and the number of graph nodes in Figure 26. The run time decreases with increasing numbers of obstacles and increases with increasing numbers of graph nodes.

Conclusions

The previous report [1] introduced a path-generation algorithm to create a safe path through a close-proximity environment for a 3-DoF manipulator arm. The algorithm has been reformulated here to improve its capability, particularly by reducing the calculation time in complex environments. This has been achieved in two ways: by reducing the calculation time in general and by changing the relationship between the calculation time and number of obstacles from exponential to logarithmic. This improves the efficiency of the algorithm as the complexity of the environment increases because the increase in calculation time per obstacle declines as the number of obstacles in the environment is increased. The general improvement in the algorithm has increased the environmental complexity that the algorithm can handle by significantly reducing the time taken to calculate a path through an environment with larger sets of sensor data. Given that the calculation time is in the order of seconds to tens of seconds for manipulator arms with similar dynamic properties in environments that do not feature fast-moving obstacles (such as those used in nuclear reactor maintenance or IED disposal missions) the calculation time range can be considered to allow real-time path generation.

Declaration of conflicting interests

The authors declare that there is no conflict of interest.

Funding

This work was supported by Engineering and Physical Sciences Research Council and Allen Vanguard via The Knowledge Transfer Network.

References

[1] D. Galvão Wall, J. Economou, H. Goyder, K. Knowles, P. Silson and M. Lawrance, "Mobile robot arm trajectory generation for operation in confined environments," *Proceedings of the Insitution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering,* vol. 229, no. 3, pp. 215-234, 2015.

[2] L. Cousineau and N. Miura, Construction Robots: The Search for New Building Technoology in Japan, ASCE Publications, 1998.

[3] R. M. Alqasemi, E. J. McCaffrey, K. D. Edwards and R. V. Dubey, "Analysis, Evaluation and Development of Wheelchair-Mounted Robotic Arms," in *IEEE International Conference on Rehabilitation Robotics*, Chicago, Illinois, 2005.

[4] Y. G. Liu and G. Nejat, "Multirobot Cooperative Learning for Semiautonomous Control in Urban Search and Rescue Applications," *Journal of Field Robotics,* vol. 33, no. 4, pp. 512-536, 2016.

[5] M. Guamieri, P. Debenesr, T. Inoh, E. Fukushima and S. Hirose, "Development of Helios VII: an arm-equipped tracked vehicle for search and rescue operations," in *IEEE International Conference on Intelligent Robots and Systems*, Sendai, Japan, 2004.

[6] M. E. Hall and R. M. Reddy, "Should every medical student receive exposure to robotic surgery?," *JOURNAL OF ROBOTIC SURGERY,* vol. 11, no. 3, pp. 375-376, 2017.

[7] Z. Fu, S. Luo and W. Zeng, "Research and Application of Radiation Resistant Robot for Nuclear Reactor," in *INTERNATIONAL CONFERENCE ON COMPUTER, MECHATRONICS AND ELECTRONIC ENGINEERING (CMEE 2016)*, Beijing, Peoples Republic of China, 2016.

[8] A. Bin Motaleb and M. B. H. M. A. Hoque, "Bomb disposal Robot Discarding explosive through wireless controlled method," in *INTERNATIONAL CONFERENCE ON INNOVATIONS IN SCIENCE, ENGINEERING AND TECHNOLOGY (ICISET 2016)*, Chittagong, Bangladesh, 2016.

[9] M. Yao, "Mathematics for Inverse Kinematics," 2009.

[10] D. R. Raj, I. J. Raglend and M. D. Anand, "Inverse Kinematics Solution of a Five Joint Robot Using Feed Forward and Elman Network," in *IEEE INTERNATIONAL CONFERENCED ON CIRCUITS, POWER AND COMPUTING TECHNOLOGIES (ICCPCT-2015)*, Nagercoil, India, 2015.

[11] S. R. Buss, "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods," *journal of graphics, gpu, and game tools,* vol. 10, no. 3, pp.

37-49, 2005.

[12] P. Srisuk, A. Sento and Y. Kitjaidure, “Inverse Kinematics Solution using Neural Networks from Forward Kinematics Equations,” in *IEEE 9TH INTERNATIONAL CONFERENCE ON KNOWLEDGE AND SMART TECHNOLOGY (KST)* , Pattaya, Thailand, 2017.

[13] M. Crenganis, R. Breaz, G. Racz and O. Bologa, “Inverse Kinematics of a 7 DOF Manipulator Using Adaptive Neuro-Fuzzy Inference Systems,” in *IEEE 12TH INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION, ROBOTICS & VISION (ICARCV)* , Guangzhou, Peoples Republic of China, 2012.

[14] G. Q. Dong and Z. H. Zhu, “Position-based visual servo control of autonomous robotic manipulators,” *Acta Astronautica,* vol. 115, pp. 291-302, 2015.

[15] T. Nakamara, “Real-Time 3-D Path Generation Method for a Robot Arm by a 2-D Dipole Field,” in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Wollongong, Australia, 2013.

[16] S. Lee, S. Baek and J. Kim, “Arm Trajectory Generation Based on RRT* for Humanoid Robot,” *ROBOT INTELLIGENCE TECHNOLOGY ANDAPPLICATIONS 3,* vol. 345, pp. 373-383, 2015.

[17] A. Hornung, K. M. Wurm, M. Bennewitz, S. C. and W. Burgard, “OctoMap: an efficient probabilistic 3D mapping framework based on octrees,” *Autonomous Robots,* vol. 34, no. 3, pp. 189-206, 2013.

[18] J. Zhao, L. Han, L. Wang and Z. Yu, “The Fuzzy PID Control Optimized by Genetic Algorithm for Trajectory Tracking of Robot Arm,” in *PROCEEDINGS OF THE 2016 IEEE 12TH WORLD CONGRESS ON INTELLIGENT CONTROL AND AUTOMATION (WCICA)* , Guilin, Peoples Republic of China, 2016.

[19] D. King, “Space servicing: past, present and future.,” in *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space: i-SAIRAS*, 2001.

[20] A. Ataka, P. Qi, H. Liu and K. Althoefer, “Real-time planner for multi-segment continuum manipulator in dynamic environments.,” in *IEEE Internation Conference on Robotics and Automation*, 2016.

[21] D. J. Galvão Wall, “PhD Thesis: A Graph-Theory-Based C-Space Path Planner for Mobile Robotic Manipulators in Close-Proximity Envorinments,” Cranfield University, 2016.

[22] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[23] N. Biggs, E. K. Lloyd and R. J. Wilson, Graph Theory, 1736-1936, Oxford University Press, 1976.

[24] H. Walther, *Ten applications of graph theory, volume 7,* Springer Science and Business Media,

2012.

[25] G. P. Kladis, J. T. Economou, K. Knowles, A. Tsourdos and B. A. White, “Aerospace Energy Conservation Utilizing Optimum Methods,” in *IEEE Vehicle Power and Propulsion Conference*, Harbin, China, 2008.

[26] J.-S. Park and S.-J. Oh, “A new concave hull algorithm and concaveness measure for n-dimensional datasets.,” *Journal of Information science and engineering,* vol. 28, no. 3, pp. 587-600, 2012.

[27] S. Lahouar, S. Zeghloul and L. Romdhane, “Path Planning for Manipulator Robots in Cluttered Environments,” in *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, 2005.

[28] S. Klanke, D. Lebedev, R. Haschke, J. Steil and H. Ritter, “Dynamic Path Planning for a 7-DOF Robot Arm,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, 2006.