Robotica (2011) volume 29, pp. 295–315. © Cambridge University Press 2010
doi:10.1017/S0263574709990786

Algorithmic path planning of static robots in three dimensions using configuration space metrics

Debanik Roy*

A 39 Vishram Tower, Shreenagar, Sector 9, Thane (West) 400604, Maharashtra, India

(Received in Final Form: December 2, 2009. First published online: April 16, 2010)

SUMMARY

Collision-free path planning for *static* robots is a demanding manifold of contemporary robotics research, vastly due to the growing industrial applications. In this paper, a novel 'visibility map'-based heuristic algorithm is used to generate near-optimal *safe* path for a three-dimensional congested robot workspace. The final path is obtainable in terms of joint configurations, by considering the *Configuration Space* of the task space. The developed algorithm has been verified initially by considering representative 2D workspaces, cluttered with different obstacles with regular geometries and then after with the spatial endeavour. A case study reveals the effectiveness of the developed modules of the configuration space mapping, pertaining to a five degrees-of-freedom low payload articulated robot.

KEYWORDS: Configuration space; Robot; Path planning; Algorithm; Collision avoidance.

1. Introduction

Obstacle avoidance and robot path planning problems have gained sufficient research attention due to its indispensable application demand in manufacturing vis-à-vis material handling sector, such as picking and placing an object, loading/unloading a component to/from a machine or storage bins. Visibility map in the configuration space (*c-space*) has become reasonably instrumental towards solving robot path planning problems and it certainly edges out other techniques widely used in the field of motion planning of robots (e.g. Voronoi Diagram, Potential Field, Cellular Automata) for *unstructured environment*. The c-space mapping algorithms, referred in the paper, are discussed with logic behind their formulation and their effectiveness in solving path planning problems under various conditions imposed *a-priori*. The visibility graph (*v-graph*) based path planning algorithm generates the equations to obtain the desired joint parameter values of the robot corresponding to the *i*th intermediate location of the end effector in the collision-free path. The developed c-space models have been verified by considering first a congested workspace in 2D and subsequently with the real spatial manifolds, cluttered with different objects. New lemma has been proposed for generating c-space maps for higher dimensional robots, e.g. having degrees-of-freedom more than three. A test case has been analysed wherein a seven degrees-of-freedom revolute robot is used for articulation, followed by a case-study with a five degrees-of-freedom articulated manipulator (RHINO XR-3) amidst an in-door environment. Both the studies essentially involve new c-space mapping thematic in higher dimensions.

Tomas Lozano Perez' postulated the fundamentals of c-space approach and proved those successfully in spatial path planning of robotic manipulators in an environment congested with polyhedral obstacles using an explicit representation of the manipulator configurations that would bring about a collision eventually.[18] However, his method suffers problem when applied to manipulators with revolute joints. In contrast to rectilinear objects, as tried by Perez', collision-avoidance algorithm in 2D for an articulated two-link planar manipulator with circular obstacles have been reported also.[15] The paradigm of *automatic transformation* of obstacles in the c-space and thereby path planning is examined with finer details,[8] such as *friction* between the obstacles.[9] Novel c-space computation algorithm for convex planar algebraic objects has been reported,[17] while *slicing* approach for the same is tried for curvilinear objects.[26,27] Nonetheless, various intricacies of the global c-space mapping techniques for a robot under static environment have been surveyed to a good extent.[33] Although the theoretical paradigms of c-space technique for solving *find-path* problem have been largely addressed in the above literature vis-à-vis a few more,[4,7,12,19,23] the bulging question of tackling collision detection under a typical manufacturing scenario, cluttered with real-life multi-featured obstacles remains largely unattended.

Survey reports on motion planning of robots in general, have been presented, with special reference to path planning problems of lower dimensionality.[13,29] The find-path problem under sufficiently-cluttered environment has been studied with several customised models, such as using distance function,[11] probabilistic function,[14] time-optimised function,[30] shape alteration paradigms[20] and sensorised stochastic method.[1] Even, novel *path transform function* for guiding the search for find-path in 2D is reported,[35] while the same for manipulators with higher degrees-of-freedom is also described.[22] All these treatises are appreciated from the context of theoretical estimation, but lacks in simulating all kinds of polyhedral obstacles.

* Corresponding author. E-mail: debanik@yahoo.com
Present address: Division of Remote Handling & Robotics, Bhabha Atomic Research Centre, Mumbai 400085, India; E-mail: deroy@barc.gov.in

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

Based on the c-space mapping, algorithmic path planning in 2D using *visibility* principle is studied,[10] followed by exhaustive theoretical analysis on visibility maps.[6] However, issues regarding computational complexity involved in developing a typical v-graph, which is $O(n^2)$, 'n' being the total number of vertices in the map, is analysed earlier.[32] The concept of *M-line* (Mean Line, as referred in the literature concerning the visibility graph-based path planning of robots.) and its uniqueness in generating near-optimal solutions against heuristic-based search algorithms has also been examined.[21]

Several researchers have reviewed the facets of path planning problem in a typical spatial manifold. A majority of these models are nothing but extrapolation of proven 2D techniques in 3D space[16,28,34] However, new methods for the generation of c-space in such cases (i.e. spatial) have been exploited too.[2,5,31] Customised solution for rapid computation of c-space obstacles has been addressed,[3] using geometric properties of collision detection between *known* static obstacles and the manipulator body, while sub-space method is being utilised in this regard.[24] The usefulness of several new algorithms using v-graph technique has been demonstrated in spatial robotic workspace.[25]

It may be mentioned at this juncture with reference to the citations above, that, although celebrated, a distinct methodology of using c-space mapping for higher dimensional robots as well as in spatial workspace is yet to be tuned. Our approach essentially calls on this lacuna of the earlier researches. We proclaim our novelty in adding new facets to the problem in a generic way, like: (a) *rationalising* c-space mapping for *higher dimensional* (e.g. 7 or 8 degrees-of-freedom) *robots*; (b) *preferential selection* of joint variables for c-space plots in 2D; (c) extension of 2D path planning algorithm in 3D through *slicing technique* (creation, validation & assimilation of c-space slices) and (d) *searching* collision-free path in 3D, using novel *visibility map-based algorithm*.

The paper has been organised in six Sections. The facets of our proposition towards c-space maps in 3D are discussed in the next Section. Section 3 delineates the c-space mapping algorithms, with the logistics and analytical models. The features of the path planning metrics and the algorithm in particular, using the concept of v-graph, have been reported in Section 4. Both 2D and spatial workspaces have been postulated, with an insight towards the analytical modelling, in respective cases alongwith test results. Section 5 presents the case study of robot path planning. Finally Section 6 concludes the paper.

2. Configuration Space Map in 3D Space: Our Proposition

2.1. Modelling of robot workspace

The robotic environment is modelled through *discretisation* of the 3D space into a number of 2D planes (Cartesian workspace), corresponding to a finite range of *waist* /base rotation of the robot. (Base rotation is earmarked for articulated robots, whereas suitable angular divisions of the entire planar area, i.e. 360° are considered for robots with non-revolute joints, e.g. prismatic.) Thus, modelling has been attempted with the sectional view of the obstacles in 2D plane.

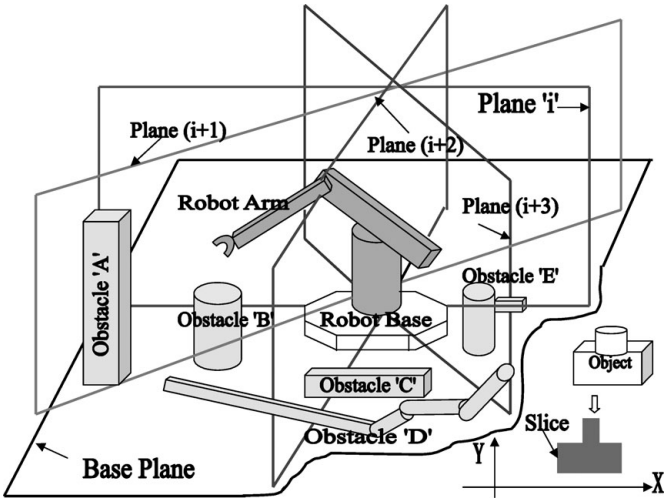


Fig. 1. Schematic view of 'Slicing Model' for a congested spatial robotic environment.

The obstacles are considered to be *regular*, i.e. having finite shape and size with standard geometrical features with known vertices in Cartesian co-ordinates. For example, obstacles with shapes such as cube, rectangular paralleopiped, trapezoid, sphere, right circular cylinder, right pyramid, etc. have been selected (as primitives) for modelling the environment. A complex obstacle has been modelled as a Boolean combination of these primitives, to have polygonal convex shape preferably. However, concave obstacles can also be used in the algorithms by approximating those to the nearest convex shapes, after considering their 'convex hulls' (polytones). Irregular-shaped obstacles have also been modelled by considering their envelopes to be of convex shapes. Circular obstacles have been approximated to the nearest squares circumscribing the original circles, thereby possessing *pseudo-vertices*.

Features of the developed technique, namely, 'Slicing Method', are: (i) alongwith shoulder, elbow and wrist (pitch only) rotations, waist rotation of the robot is considered, which is guided by a finite range vis-à-vis a finite resolution; (ii) the entire 3D workspace is divided into a number of 2D planes, according to the number of 'segments' of the waist rotation; (iii) for every fixed angle of rotation of the waist, a 2D plane is to be constructed, where all other variables like shoulder, elbow and wrist pitch movements are possible; (iv) corresponding to each of the 2D slices of the workspace, either one obstacle entirely or a part of it will be generated, depending on the value of the resolution chosen for waist rotation.

Corresponding to each slice, one c-space map (considering only two joint variables at a time) is to be developed and likewise, several maps will be obtained for all the remaining slices. The final combination of the colliding joint variable values will be the union of all those sets of the values for each slice. Nevertheless, the process of computation can be simplified by taking some finite number of slices, e.g. four to five planes. Figure 1 pictorially illustrates the

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

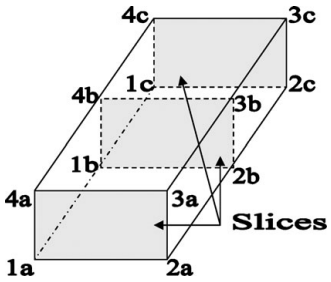


Fig. 2. Node numbering thematic for sliced obstacles.

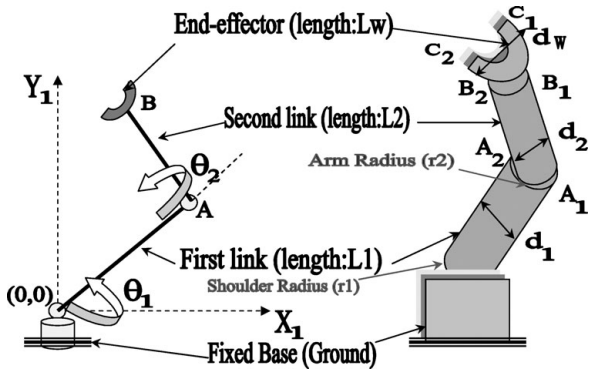


Fig. 3. Representative schematics of a two-link manipulator with revolute joints.

above-mentioned postulation, wherein the robotic workspace consists of various categories of obstacles, like regular geometry (e.g. obstacle 'A', 'B' & 'C') and integrated geometry, where Boolean combination of regular geometries is considered (e.g. obstacle 'D' & 'E').

By knowing the intercept co-ordinates for each slice, an equivalent 2D *slice workspace* is developed; simply by projecting those intercept lines. Obviously the height of the obstacle will now play a vital role and the projecting length will be equal to the height of the obstacle. In a similar manner, obstacles with varying height (say, slant-top type) can also be considered, with differential length of projection. As before, irregular-shaped obstacles can be approximated by means of some standard regular shape, either uniform or varying height, using the same model.

The total number of c-space plots for the entire cluttered environment will depend on the number of slices each obstacle has and also thereby the average number of slices obtained. Since each of those sliced c-space will represent obstacle geometries, fully or partially, it is important to label the nodes of the obstacle-slices so produced. In general, if a particular obstacle has got 'k' slices, having 'n' nodes each, then a generalised node of that obstacle will be labeled as, 'n_k'. However, to avoid ambiguity, 'k' is considered alphabetic only. Figure 2 shows a representative obstacle with slices, where node '3_b' signifies the third node of the bth slice, which is incidentally the second slice of the obstacle.

2.2. Paradigms of the c-space mapping algorithms in 2D sliced *workspace*

In the present work, *sliced* c-space maps have been generated considering a two degrees-of-freedom revolute type manipulator having finite dimensions (refer Fig. 3) and an environment cluttered with polygonal obstacles. Both manipulator links and obstacles are represented as convex or concave polygons. (Concave obstacles are modeled as a combination of several convex polygons.)

The obstacles are considered to be *regular* in shape with fixed dimensions, having a well-defined shape (sectional view) in 2D, preferably convex, for easier calculations. This has been made purposefully as in most of the manufacturing and/or shop-floor activities *geometric* objects are being handled by the robot, for example, loading and unloading of components to/from the machine, handling of semi-finished components between machines, storage and retrieval of finished components into bins etc. The philosophy behind these mapping algorithms is to consider each complex obstacle as a boolean combination of various primitives, *viz.*, '*Point*', '*Line*' and '*Circle*'. That is to say, if the obstacle is theoretically considered as a 'point', 'line' or 'circle' in shape in 2D, then colliding angle of the robot link(s) with those will be obtained and c-space maps can be drawn there from. These algorithms can also be applied for concave objects by considering the 'convex hull' of those and proceeding in the same manner taking that as the *new* obstacle. Similarly, irregular shaped objects can also be tackled with these models, in which *envelope* of the object is to be considered to get the nearest convex shape. Obviously accuracy of the results will suffer to some extent by this approximation, but it would be a reasonable solution for practical situations.

3. Details of the c-Space Mapping Algorithms Developed

3.1. Overview of the mapping algorithms

3.1.1. c-space transformation of 'point'. This algorithm gives the c-space data for an obstacle, considered theoretically as a *point* in Cartesian space. The robot with line representation is being considered here. (refer Fig. 3).

Algorithm:

1. Input robot base co-ordinates, link lengths, the specified *point*, range of joint angles & angular resolution.
2. Calculate the distance of the *point* from robot base.
3. Initialise the loop with iteration $i = 1$.
4. Check whether the first link is colliding.
5. Calculate the positional details of the first link (i.e. under colliding conditions), if step 4 is true.
6. Check the collision with the second link, if step 4 is false.
7. Calculate the colliding details of the second link, if step 6 is true.
8. Output the colliding angles for suitable cases.
9. $i = i + 1$.
10. Continue till all combinations of joint angles are checked.

3.1.2. c-space transformation of 'line'. Here the obstacle has been considered as a regular polygon, bounded by several straight line segments, as 'edges'. The algorithm is valid for obstacles having rectangle, square, triangle, trapezium etc. shaped cross-section in the vertical plane.

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

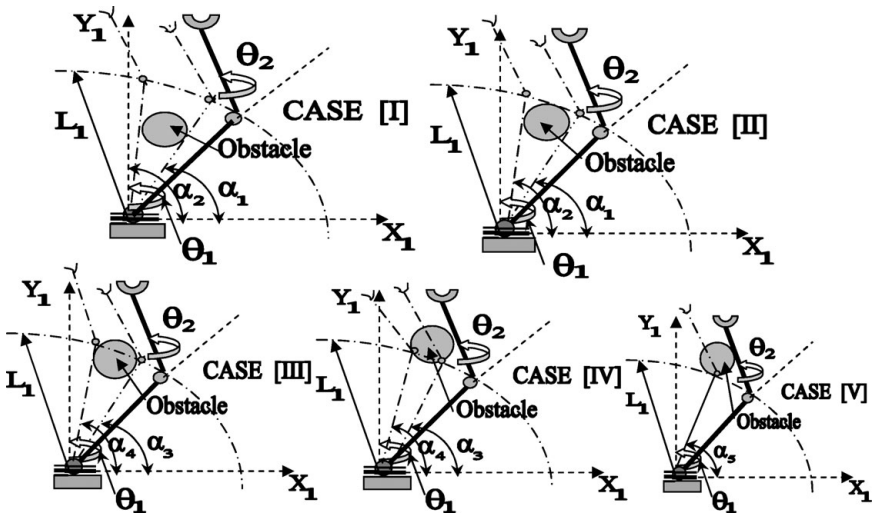


Fig. 4. Various possible colliding combinations of a two-link robot with circular obstacle.

3.1.3. c-space transformation of 'circle'. This algorithm for collision detection tackles the spherical obstacles with circular cross-section in the vertical plane. Various possible colliding conditions of the circular obstacle with the robot link(s) are depicted in Fig. 4. It depends entirely on the location of the obstacle(s) with respect to the location of the robot in the workspace.

Figure 4 shows five different cases of collision, considering only the first link. However, the paradigm is valid for subsequent links also. These cases are:

Case 1: Obstacle is fully within the range of the first link.
Case 2: Obstacle is within the first link's range, touching the *range circle* internally.
Case 3: Obstacle is collidable by the first link, with its centre inside the range of the first link.
Case 4: Same as before, but centre is outside the range of the first link.
Case 5: Obstacle is touching the range circle of the first link externally.

3.1.4. c-space transformation with finite dimensions of the robot. Modifications of the previous algorithms for *POINT*, *LINE* and *CIRCLE* obstacles have been considered with finite dimensions of the robot arms (refer Fig. 3). Also, planar three-link manipulator is considered now, where the third link is nothing but the end effector. Since planar movements are taken into account, only pitch motion of the wrist is considered along with shoulder and elbow rotations. The exhaustive list of input parameters for these cases will be as follows, *viz.* (i) Robot base co-ordinates (x_b, y_b); (ii) Length of the upper arm or shoulder (l_1); (iii) Length of the fore arm or elbow (l_2); (iv) Length of the end effector or *wrist* (l_w); (v) Width of the upper arm (d_1); (vi) Width of the fore arm (d_2); (vii) Width of the end effector (d_w); (viii) Radii of curvature for upper and fore arm (r_1 & r_2 respectively); (ix) The co-ordinates of the *point*: (x_1, y_1) or *line*: [(x_1, y_1) & (x_2, y_2) as end points] or *circle*: [centre at (x_c, y_c) & radius: 'r'].

3.2. Analytical model of the mapping algorithms

The analytical models of various c-space mapping algorithms, described earlier, have been grouped into two categories, *viz.* (a) Model for *LINE* obstacles and (b) Model for *CIRCLE* obstacles. These are being described below.

3.2.1. Model for line obstacles. The ideation of collision detection phenomena for *Line* obstacle is based on the intersection of the line segments in 2D considering only the kinematic chain of the manipulator. The positional information, i.e. co-ordinates (x_k, y_k) of the manipulator joints can be generalised as,

$$[x_k, y_k]^T = [x_b, y_b]^T + \left[\sum_{k=1}^{k=n} l_k \cos\left(\sum_{j=1}^{i=k} \theta_j\right), \sum_{k=1}^{k=n} l_k \sin\left(\sum_{j=1}^{i=k} \theta_j\right)\right]^T \quad \forall k = 1, 2, 3, \ldots\ldots\ldots\ldots, n \;\&\; \forall j = 1, 2, \ldots\ldots\ldots, k, \tag{1}$$

where, $[x_b, y_b]^T$: Robot base co-ordinate vector in Newtonian frame of reference; l_k: kth link length of the manipulator; θ_j: jth joint angle of the manipulator.

The slope of the *line* (i.e. the edge of the obstacle) with (x_1, y_1) & (x_2, y_2) as end points is given by,

$$m_0 = |(y_2 - y_1)|/(x_2 - x_1)| = |\Delta y/\Delta x|. \tag{2}$$

Hence, the co-ordinates of the intersection point between the obstacle edge(s) and the robot link(s) are

$$x_{\text{int}_k} = (y_k - y_1 - m_k x_k + m_0 x_1)/(m_0 - m_k), \tag{3a}$$

$$y_{\text{int}_k} = [m_0 m_k(x_k - x_1) - m_0 y_k + m_k y_1]/(m_k - m_0), \tag{3b}$$

where, $[x_{\text{int}_k}, y_{\text{int}_k}]^T$: The intersection point vector for the kth robot link with the *line* and m_k: The slope of the kth robot link.

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

For a three-link planar revolute manipulator, we have, therefore,

$$m_1 = \tan\theta_{1i}, \quad \text{where } \theta_{1i} \in [\theta_{1_\min}, \theta_{1_\max}] \text{ and}$$
$$m_2 = \tan(\theta_{1i} + \theta_{2j}), \quad \text{where } \theta_{2j} \in [\theta_{2_\min}, \theta_{2_\max}]$$
$$\text{with } \theta_{1i} \text{ as defined earlier.}$$

The above model can be extended likewise, considering finite dimension of the manipulator, i.e. having widths of the arms, *viz.* $\{d_k\}$.

Nonetheless, considering finite dimensions, the generalised co-ordinates of the manipulator joints can be evaluated from,

$$[x_{Ak}, y_{Ak}]^T = [x_b, y_b]^T + \left[\sum_{k=1}^{k=n} l_k \cos\left(\sum_{j=1}^{i=k}\theta_j\right), \sum_{k=1}^{k=n} l_k \sin\left(\sum_{j=1}^{i=k}\theta_j\right)\right]^T + \left[d_k/2 \sin\left(\sum_{j=1}^{i=k}\theta_j\right), -d_k/2 \cos\left(\sum_{j=1}^{i=k}\theta_j\right)\right]^T \quad \forall k = 1, 2, 3, \ldots\ldots\ldots\ldots, n \;\&\; \forall j = 1, 2, \ldots\ldots\ldots, k \tag{4}$$

and

$$[x_{Bk}, y_{Bk}]^T = [x_b, y_b]^T + \left[\sum_{k=1}^{k=n} l_k \cos\left(\sum_{j=1}^{i=k}\theta_j\right), \sum_{k=1}^{k=n} l_k \sin\left(\sum_{j=1}^{i=k}\theta_j\right)\right]^T + \left[-d_k/2 \sin\left(\sum_{j=1}^{i=k}\theta_j\right), d_k/2 \cos\left(\sum_{j=1}^{i=k}\theta_j\right)\right]^T \quad \forall k = 1, 2, 3, \ldots\ldots\ldots\ldots, n \;\&\; \forall j = 1, 2, \ldots\ldots\ldots, k. \tag{5}$$

Hence, the co-ordinates of the intersecting point(s) between the robot arm(s) and the *edges* of the obstacle(s) become,

$$x_{\text{int}A_k} = (y_{Ak} - y_1 - m_k x_{Ak} + m_0 x_1)/(m_0 - m_k); \tag{6a}$$
$$y_{\text{int}A_k} = [m_0 m_k(x_{Ak} - x_1) - m_0 y_{Ak} + m_k y_1]/(m_k - m_0); \tag{6b}$$
$$x_{\text{int}B_k} = (y_{Bk} - y_1 - m_k x_{Bk} + m_0 x_1)/(m_0 - m_k); \tag{6c}$$
$$y_{\text{int}B_k} = [m_0 m_k(x_{Bk} - x_1) - m_0 y_{Bk} + m_k y_1]/(m_k - m_0); \tag{6d}$$

considering two possible collisions per arm at the most.

3.2.2. Model for circle obstacles. With reference to Fig. 4, let the following nomenclatures be defined,

(x_b, y_b): Robot base co-ordinates in Newtonian frame;

l_1 & l_2: Lengths of the first and second link of the robot respectively;

(x_c, y_c): Co-ordinates of the centre of the 'circle' obstacle and

r: Radius of the 'circle'.

Let,

$$d = [(x_c - x_b)^2 + (y_c - y_b)^2]^{1/2}. \tag{7}$$

If collision is detected with the first link of the robot, then the range of colliding angles for case I & II (i.e. α_1 & α_2) and for case III & IV (i.e. α_3 & α_4) will be evaluated as shown below,

$$\alpha_{2,1} = \tan^{-1}[(y_c - y_b)/(x_c - x_b)] \pm \tan^{-1}[r/(d^2 - r^2)^{1/2}] \tag{8}$$

$$\alpha_{4,3} = \tan^{-1}[(y_c - y_b)/(x_c - x_b)] \pm 2\tan^{-1}[(s - l_1)(s - d)/s(s - r)^{1/2}] \tag{9}$$

where,

$$2s = (l_1 + d + r) \tag{10}$$

Obviously, only one colliding angle, *viz.* α_5 will be obtained with case *V*, i.e.

$$\alpha_5 = \tan^{-1}[(y_c - y_b)/(x_c - x_b)] \tag{11}$$

When the collision occurs with the second link, i.e. all the values of the first joint angle are collidable, the ranges for collision will be obtained as given below.

The co-ordinates of the start point of the second link, which serves as the *instantaneous base* of the robot, are given by,

$$[x_b', y_b']^T = [x_b,\ y_b]^T + [l_1 \cos\theta_1,\ l_1 \sin\theta_1]^T. \tag{12}$$

The colliding range for the second link is between θ_{2S} and θ_{2f} against case I & II, where,

$$\theta_{2S} = \beta_1 - \theta_{1i} \quad \text{and} \quad \theta_{2f} = \beta_2 - \theta_{1i}, \quad \forall\theta_{1i}, \text{ where } \theta_{1_\min} \le \theta_{1i} \le q_{1_\max} \tag{13}$$

and

$$\beta_{2.1} = \tan^{-1}[(y_c - y_b)/(x_c - x_b)] \pm \tan^{-1}(r/p) \tag{14}$$

where,

$$p = (h^2 - r^2)^{1/2} \tag{15}$$

and

$$h = [(x_c - x_b')^2 + (y_c - y_b')^2]^{1/2}. \tag{16}$$

For case III & IV, the colliding range will be θ_{3S}–θ_{3f}, which can be evaluated as,

$$\theta_{3S} = \beta_3 - \theta_{1i} \text{ and } \theta_{3f} = \beta_4 - \theta_{1i}, \quad \forall\theta_{1i}, \text{ where } \theta_{1i} \in [\theta_{1_\min}, \theta_{1_\max}] \tag{17}$$

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

Table I. Technical features of the robot under consideration.

Type of robot considered	Revolute
No. of links	2
No. of degrees-of-freedom	2
Length of the links	First link: 5 units; second link: 4 units
Co-ordinates of the robot base	$x_b = 5$; $y_b = 5$
Ranges of rotation of the joints (Anti-clockwise)	Case I: For non-circular polyhedral obstacles Joint angle 1: 0–360°. Joint angle 2: 0–260°. Case II: For circular obstacles Joint angle 1: −40 to 240°. Joint angle 2: −180 to 180°
Resolution of Joint Rotation	5° (for both joints)

and

$$\beta_{4,3} = \tan^{-1}[(y_c - y'_b)/(x_c - x'_b)] \pm 2\tan^{-1}[(s' - l_2)(s' - h)/s'(s' - r)]^{1/2} \quad (18)$$

where,

$$2s' = (l_2 + h + r) \quad (19)$$

For case *V*, the particular formidable value of the joint angle is,

$$\theta_4 = \tan^{-1}[(y_c - y'_b)/(x_c - x'_b) - \theta_{1i}. \quad (20)$$

The above equations need to be altered for collision checking with finite link dimensions of the manipulator, considering r_1 and r_2 as the radii of curvature of the upper arm and fore arm respectively (refer Fig. 3). The modifications required are: (i) 'l_1' is to be replaced by l_u ($= l_1 + r_1$); (ii) 'l_2' is to be replaced by l_f ($= l_2 + r_2$) and (iii) 'r' is to be replaced by $r_{nj} = r + d_j/2, \forall\, j = 1, 2, 3$ corresponding to collision with upper arm, fore arm and the end effector.

3.3. Illustrations using the developed algorithms

The c-space mapping algorithms have been tested with two different robot workspaces in 2D, as detailed below. Technical details of the robot under consideration for these workspaces are highlighted in Table I.

Figure 5 presents a 2D environment with non-circular polygonal obstacles with a revolute type robot located in between.

After obtaining the relevant c-space data, various plots of joint angle 1 vs. joint angle 2 are made. It is to be noted that the final data points are those, which are common values of formidable angles for the entire obstacle. Obviously, the points, which are inside or on the closed boundary of the curves, are 'collidable' combinations, and, hence, those should not be attempted for robot path planning.

Another case is studied, as shown in Fig. 6, with circular obstacles only. There is a distinct difference in the appearance in the c-space maps between obstacles when collision occurs

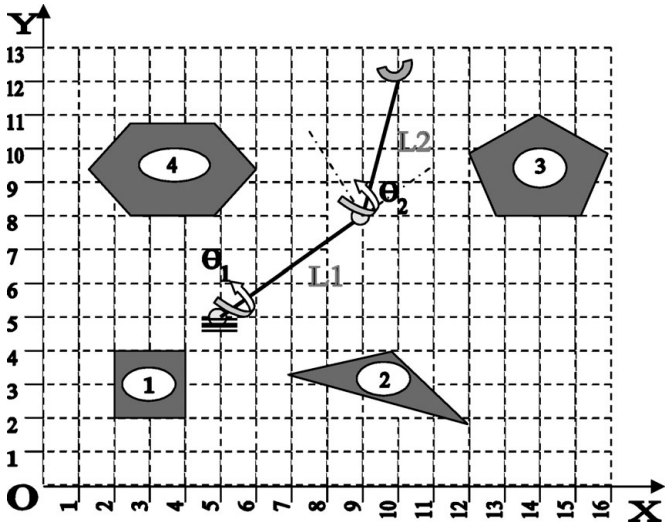


Fig. 5. A representative 2D environment congested with non-circular polygonal obstacles.

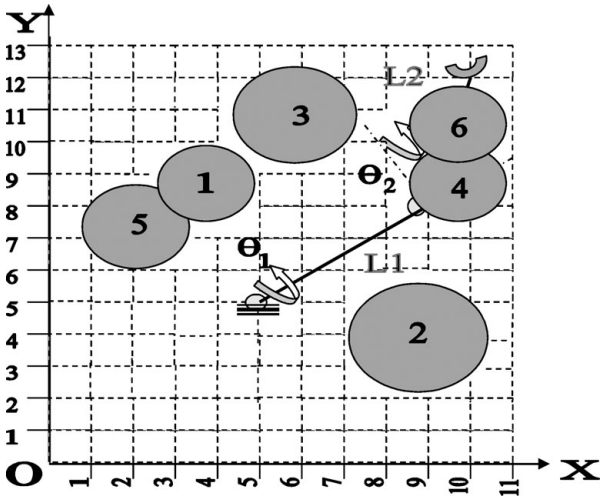


Fig. 6. A representative 2-D environment filled with circular obstacles.

with the first as constrained with the same for the second link.

Figures 7 and 8 show the final c-space for the above two environments.

3.4. c-space maps for higher dimensionality

The development of c-space is relatively simpler while we have two degrees-of-freedom robotic manipulators. As described in earlier sections, irrespective of the nature of the environment, we can generate simple planar maps, corresponding to the variations in the joint angles (in case of revolute robots). Thus the mapping between task space and c-space is truly mathematical and involves computationable solutions for inverse kinematics routines. The procedure of generating c-space can be extrapolated to three degrees-of-freedom robots at most, wherein we get 3D plot, i.e. mathematically speaking, c-space *surface*. However, this procedure can not be applied to higher dimensional robots, having degrees-of-freedom more than three. In fact, c-space surface of dimensions greater than three is unrealisable,

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

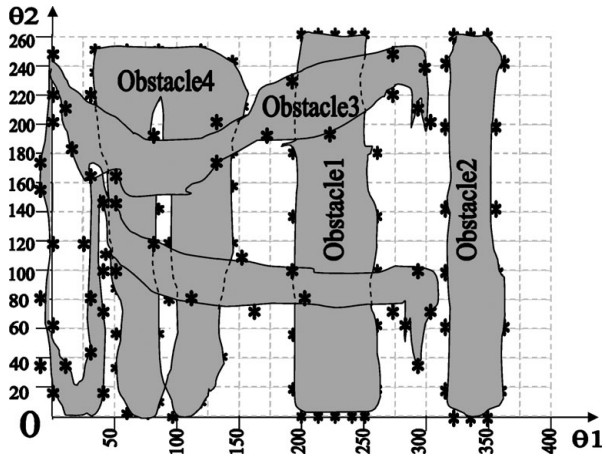


Fig. 7. Final c-space mapping for the environment filled with non-circular polygonal obstacles.

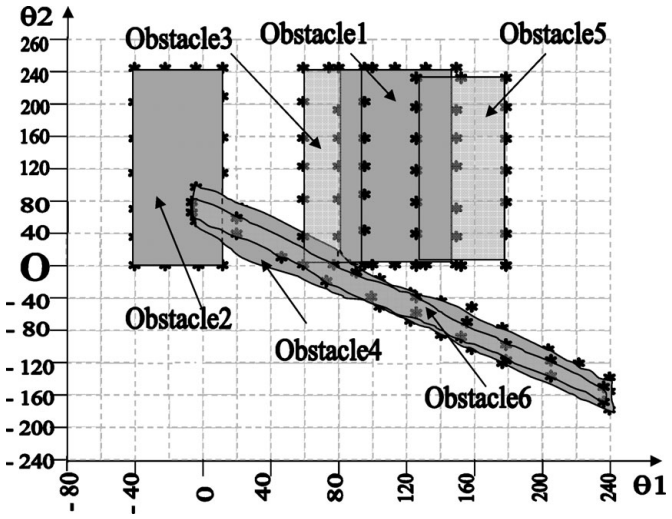


Fig. 8. Final c-space mapping for the environment congested with circular obstacles.

although it is quite common to have such robots in practice amidst cluttered environment. For example, let us take the case of a workspace for a seven degrees-of-freedom articulated robot, as depicted in Fig. 9. Here we need to consider variations in each of the seven joint angles, *viz.* θ_1, $\theta_2, \ldots\ldots, \theta_7$ (the last two degrees-of-freedoms are attributed to the wrist rotations) towards avoiding collision with the obstacles.

So, the question arises as how can we tackle this problem of realising higher dimensionality of c-space mapping. Hence it is a clear case of building *composite c-space map*, with proper characterisation of null space. We propose a model for this mapping in higher dimensionality as detailed below.

3.4.1. Lemma.

1. Identify the degrees-of-freedom of the robot [‘n’] and notify those.
2. The robot is conceptualised as a serial chain of micro-robots of two degrees-of-freedom each.
3. We will consider a total of ‘k’ planar c-space plots, where $k = n/2$ if ‘n’ is even and $k = (n + 1)/2$ if ‘n’ is odd.

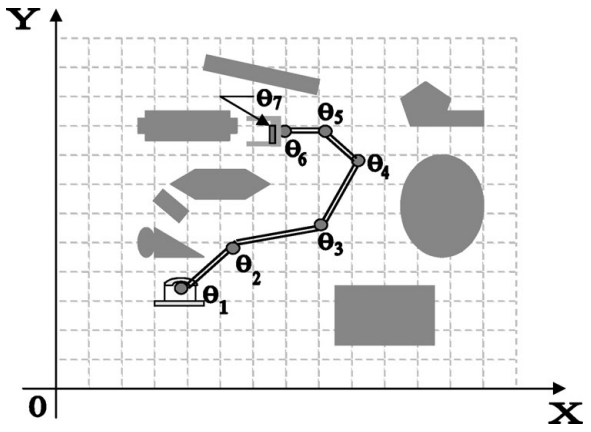


Fig. 9. A representative cluttered environment with seven degrees-of-freedom revolute robot.

4. If ‘n’ is even, then the pairs for the planar c-space plots will be: $[q_1–q_2]$, $[q_3–q_4], \ldots\ldots [q_{n-1}–q_n]$, where ‘q’ denotes the generalised joint variable of the manipulator.
5. If ‘n’ is odd, then the pairs for the planar c-space plots will be: $[q_1–q_2]$, $[q_3–q_4], \ldots\ldots [q_{n-2}–q_{n-1}]$, $[q_{n-1}–q_n]$.
6. In a way, we are considering several *virtual two-link mini manipulators,* located at the respective joints of the original manipulator.
7. Out of the plots so generated, select the *most significant* c-space map.
8. One way of accessing the most significant c-space map is to consider finite measurement of the planar area of the c-space. A larger area automatically indicates more complex dynamics of the joint variables so far as the collision avoidance is concerned.
9. Alternatively, significant c-space plots will be those having multiple disjointed loops, i.e. regions of formidable area. Individually the regions may be of smaller area, but the multiplicity of their occurrence adds complexity to the scenario.
10. Once the most significant c-space map is selected, the locations corresponding to ‘S’ and ‘G’ are to be affixed in that plot. This will be achieved using inverse kinematics routine from ‘S’(x, y) and ‘G’ (x, y).
11. For the most significant plot so obtained, all joint variables, except the two used in the plot will be *constant*. For example, if $[q_3–q_4]$ plot is the most significant one, then $q_1, q_2, \ldots\ldots q_{n-1}, q_n$ are constant except q_3 and q_4.
12. In general, if $[q_i–q_j]$ plot be the most significant, then the set $\{q_1, q_2, \ldots\ldots q_{n-1}, q_n\}$, except $[q_i–q_j]$, will be constant. And, the value of the set $\{q_1, q_2, \ldots\ldots, q_{i-1}\}$ will be ascertained by the inverse kinematic solution of ‘S’ while the other set, $\{q_{j+1}, q_{j+2}, \ldots\ldots, q_n\}$ will be determined by the inverse kinematic solution of ‘G’.

3.4.2. Schematic of the model. Let us take an example of a seven degrees-of-freedom articulated robot, similar to one illustrated in Fig. 9. According to the lemma proposed in 3.4.1, there will be four planar c-space plots, namely, $[\theta_1–\theta_2]$, $[\theta_3–\theta_4]$, $[\theta_5–\theta_6]$ and $[\theta_6–\theta_7]$. Figure 10 shows a sample view of these segmental c-space maps.

As it may be apparent from Fig. 10, either $[\theta_1–\theta_2]$ or $[\theta_3–\theta_4]$ plot can be significant, considering *larger area* or

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

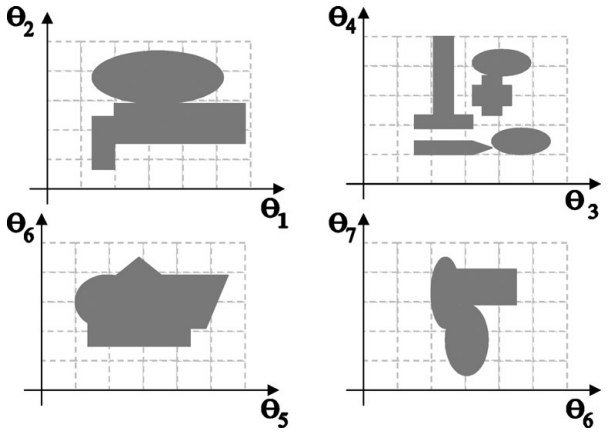


Fig. 10. Sample view of the composite C-space map for a seven degrees-of-freedom robot.

presence of multiple loops criterion (refer serial no. 8 & 9 of 3.4.1).

3.4.3. Generation of c-space plots: concept of equivalent circle. In order to compute c-space data points for any particular combination of consecutive joint variable pair for the higher dimensional robot, we would use a new concept, *viz.* the formulation of *Equivalent Circle* at the end of amidst the pair of links. Since we are considering *virtual two-link mini-manipulators* for the generation of c-space maps in pair, we would theoretically divide the links in two groups. The links, directly related to the generation of the specific c-space map, are termed as *active links*, while the others are known as *dummy links*. The philosophy of this equivalent circle is to re-represent the higher dimensional manipulator with only the active links and the joints therein, interfaced with circular zone(s) either at the bottom of the first active link or at the tip of the second active link. In general, the equivalent circles are constructed considering full rotational freedom of all the dummy links, located before /after the active links.

For example, if we wish to generate $[\theta_1$–$\theta_2]$ plot for the seven d.o.f. manipulator, then the *equivalent circle* alias *equivalent formidable zone* is to be constructed adjacent to the end the second link and circumscribing the remaining links. Figure 11 schematically presents the concept of *equivalent formidable zone*, with first two links as active links for a seven d.o.f. manipulator.

However it is possible to have two *equivalent formidable zones* in cases where some intermediate links are considered for c-space plots. For example, if the third & fourth links of the manipulator become active links, then there will be two formidable zones, as illustrated in Fig. 12. Of course, as per this proposition, there can't be more than two formidable zones for any higher dimensional manipulator.

It is essential to locate the centre of the *equivalent circle* vis-à-vis its radius (equivalent radius, $R_{eq.}$), in order to start computing for colliding combinations. However, the formulations for equivalent radii are not same in the two cases, as cited in Figs. 11 and 12. In fact, the centre of the equivalent circle, for cases wherein active links are followed by dummy links, will be at the base of the manipulator and its radius will be the summation of the lengths of the dummy links till we reach the first active link. For example, $R_{eq.1}$ in Fig. 12 will be the added sum of L1 & L2. Figure 13

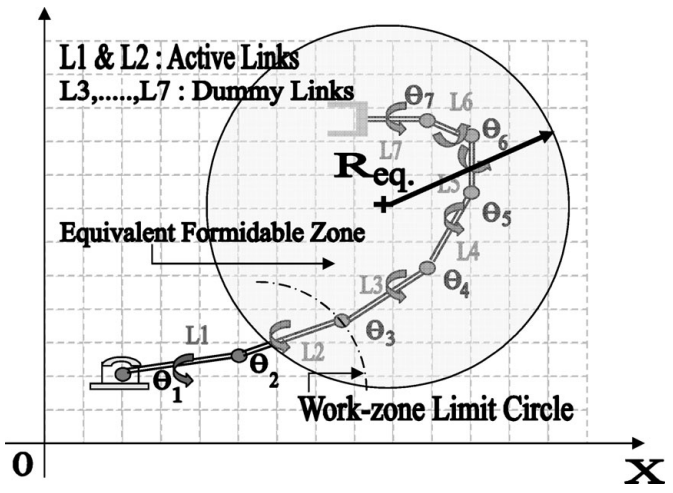


Fig. 11. Schematic view of equivalent formidable zone for a seven d.o.f. manipulator.

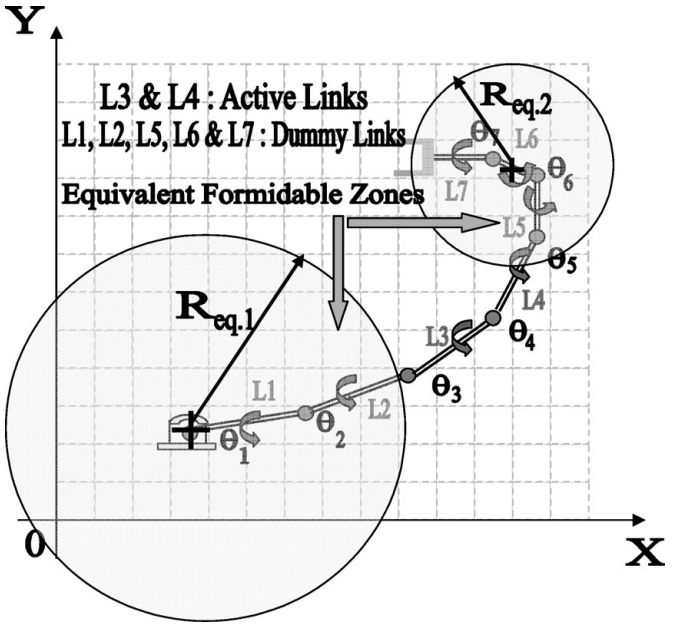


Fig. 12. Occurrence of two equivalent formidable zones for a seven d.o.f. manipulator.

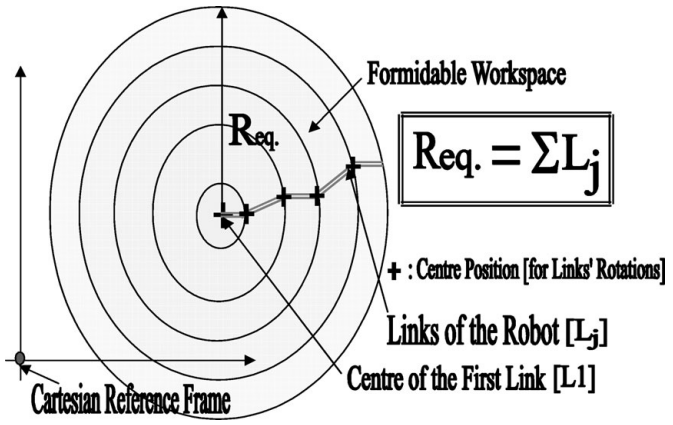


Fig. 13. Schematic showing the analytical layout for the evaluation of equivalent radius [type I].

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

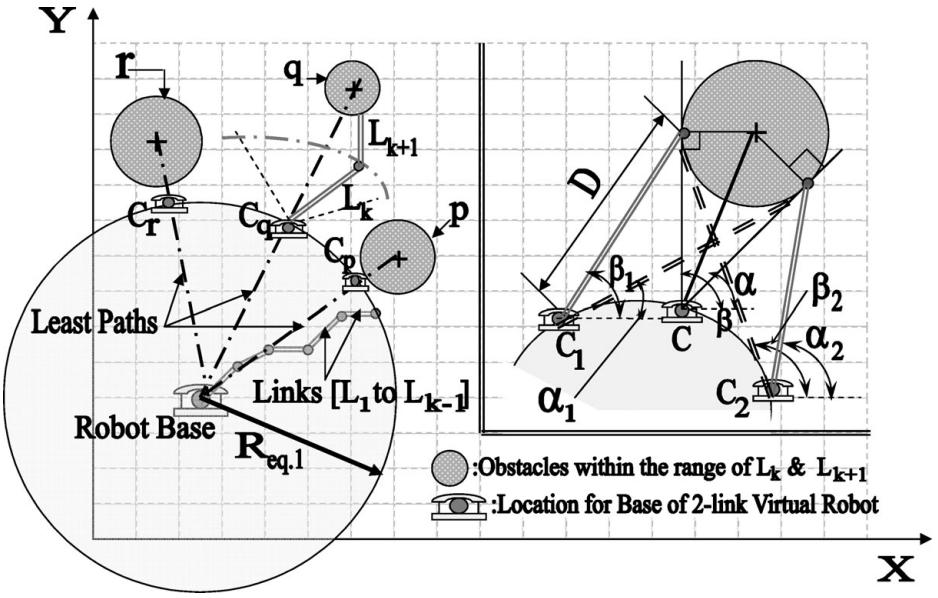


Fig. 14. Locations of the base for the two link virtual robot with respect to equivalent circle [type I].

shows the computational backup for the evaluation of the *radius* of the equivalent formidable zone/circle in this case, nomenclated as *equivalent radius* [*type I*].

Although finding centre and calculating equivalent radius [type I] is straight-forward, evaluation of the new *base* for the *virtual two-link robot* is critical. For example, as shown in Fig. 12, we need to find out the possible location of the base for the virtual robot, comprising of L3 & L4. This has been explained in Fig. 14, wherein we adopt a methodology, called *Least Path*. The least path(s) is/are the straight line segment(s) joining the centres of equivalent circle [type I] and the obstacles in the vicinity of the virtual robot. The respective locations for the base of the virtual robot will be the point of intersection of the least path and the equivalent circle. Thus, as per Fig. 14, there will be three base-points, *viz.* 'C_p', 'C_q' & 'C_r' corresponding to three obstacles, vide 'p', 'q' & 'r', which are situated within the limit-zone of the virtual robot with links L_k & L_{k+1}. However, this location of the base can range between two extreme points, as detailed in the inset of Fig. 14. Two cases may appear here, *viz.* when the obstacle is (a) within the range of both L_k & L_{k+1} and (b) outside the range of L_k. Now, considering the first link of the virtual robot is just able to touch the obstacle we can get two extreme positions for the base, e.g. 'C_1' & 'C_2'. As explained earlier, the point 'C' is the other location for the base, situated on the least path. From geometry, we can assign 'D', which is numerically equal to either L_k or L_{k+1}, depending upon which link is colliding with that obstacle. The collidable range of joint angles, corresponding to 'C', 'C_1' & 'C_2' will be (β-α), (β_1-α_1) & (β_2-α_2) respectively. Thus, in general a formidable range from α_2 to β_1 should be selected for c-space mapping in (θ_k–θ_{k+1}) plot.

In contrary to this situation, the other one, namely where dummy links are followed by active links, is more intricate so far the thematic is concerned. Figure 15 explains this case of evaluating *equivalent radius [type II]*, the corresponding circular zone being *optimised* between two extremes, *viz.*

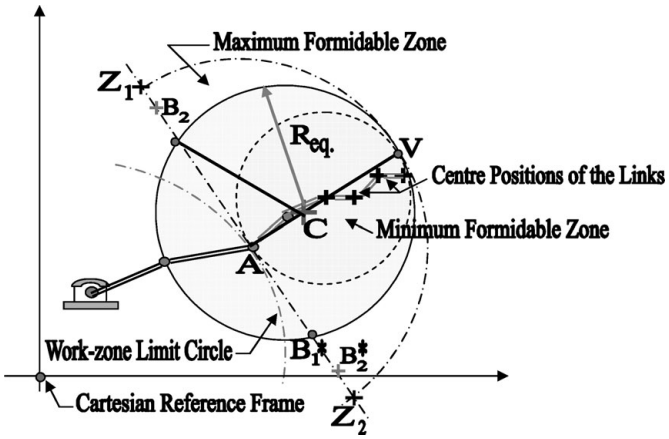


Fig. 15. Schematic showing the disposition of the equivalent formidable circle [type II].

maximum and minimum formidable zones. The minimum formidable zone is a circular space, tangent to the work-zone limit circle at 'A' while the maximum formidable zone is a semi-circular area, with two opposite extremities as 'Z_1' & 'Z_2'. Several feasible formidable zones are theoretically possible in-between, with pair of *chordal end-points* as [B_1–B_1^*] or [B_2–B_2^*] etc. It may be noted that all the three formidable zones, namely the minimum, maximum & optimum, share the common vertex 'V'.

The *equivalent radius [type II]* is evaluated using geometrical attributes, as detailed in Fig. 16. Here, the points 'A', 'C' & 'C_m' represent the locations of the centres of the maximum, equivalent & minimum formidable zones respectively. As evident from the figure, the ratio between the two line segments, *viz.* the semi-chordal length of the equivalent circle and the radius of the maximum formidable zone is 'k', where $0 < k < 1$. In-line with the numerical

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

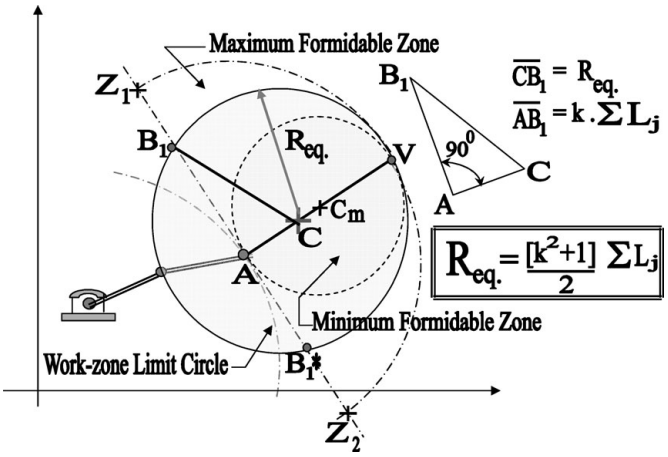


Fig. 16. Schematic showing the analytical layout for the evaluation of equivalent radius [type II].

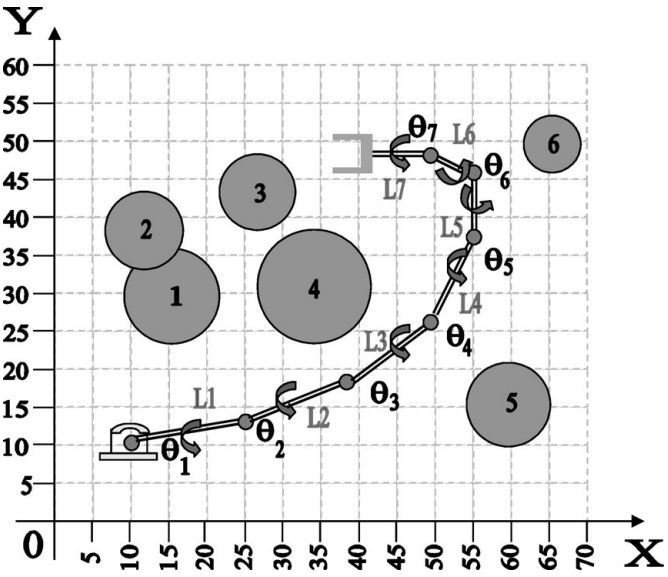


Fig. 17. Workspace layout of the seven degrees-of-freedom articulated robot.

evaluation of '$R_{eq.}$', the location of the centre, 'C' can be determined also.[1]

3.5. Evaluation of c-space plots for higher dimensional robot: example

Here we will study a specific case for a seven degrees-of-freedom robot, amidst a 2D cluttered environment as shown in Fig. 17. The technical parameters of the robot, comprising link lengths, $\{l_i, \forall i = 1,2,\ldots 7\}$ and joint limits $\{\theta_i, \forall i = 1,2,\ldots 7)$, are highlighted in Table II. Circular obstacles are considered for simplicity in computations. The locations of the respective centres and diameters of the obstacles (expressed in suitable units) are presented in Table III.

Now, in this case of seven degrees-of-freedom revolute robot, we will have four different c-space plots, namely, $[\theta_1$–$\theta_2]$, $[\theta_3$–$\theta_4]$, $[\theta_5$–$\theta_6]$ & $[\theta_6$–$\theta_7]$. All of these plots use

[1] The location of the centre is determined by evaluating the length of the line-segment, AC, which is numerically equal to $[(1-k^2)/2]\Sigma L_j$ and it is also at a distance of $(k^2/2)\Sigma L_j$ from the centre of the minimum formidable zone, i.e. 'C_m'.

Table II. Technical facets of the higher dimensional robot under consideration.

Type of robot considered	Revolute
No. of Links	7
No. of degrees-of-freedom	7
Length of the links [in suitable units]	$l_1 = 15.52$; $l_2 = 14.87$; $l_3 = 13.04$; $l_4 = 12.54$; $l_5 = 9.0$; $l_6 = 5.22$ & $l_7 = 8.0$
Co-ordinates of the robot base	$x_b = 10$; $y_b = 10$
Ranges of rotation of the joints (Anti-clockwise)	θ_1: -20° to 140°; θ_2: 0°–120°; θ_3: 0°–80°; θ_4: -5° to 90°; θ_5: 10°–50°; θ_6: -5°–55°; $-_7$: 5°–35°
Resolution of joint rotation	3° (for all joints)

Table III. Obstacle signature, as per workspace layout of Fig. 16.

Obstacle no.	Diameter	Location of centre
1	15.88	$(x = 15, y = 30)$
2	13	$(x = 12.5, y = 37.5)$
3	12.7	$(x = 27.5, y = 42.5)$
4	19	$(x = 35, y = 30)$
5	14	$(x = 60, y = 15)$
6	9.53	$(x = 65, y = 50)$

collision-detection algorithms, described earlier, for each of the obstacles separately, taking into account the concepts of *equivalent circles*. These c-space plots are illustrated in Fig. 18. A gross estimate reveal that $[\theta_1$–$\theta_2]$, $[\theta_3$–$\theta_4]$, $[\theta_5$–$\theta_6]$ & $[\theta_6$–$\theta_7]$ plots occupy a planar area measuring (160×120), (80×95), $(40, 60)$ & (60×30) sq. units respectively. It is evident from Fig. 18 that although complexity-wise both $[\theta_1$–$\theta_2]$ and $[\theta_3$–$\theta_4]$ plots are roughly at par, but the former is to be selected as the most significant c-space plot as it is also the largest in size.

It may be noted that while c-space plot for a 2 d.o.f. robot (working in 2D or 3D task space) can be composed of irregular non-geometric shapes (refer Fig. 7), the same for higher d.o.f. robots are perfectly geometric (refer Fig. 18). This is happening because of the incorporation of the concept of 'formidable zones' for higher dimensional robots, wherein we are deliberately allowing the collidable zone to engulf more regions in the c-space. In fact, in most of cases for higher d.o.f. robots, the c-space zones are perfect rectangular in shapes, between the minimum & maximum limits of the participating joint angles. For example, in Fig. 18, (θ_1 vs. θ_2) c-space slice plot the final rectangle is constituted between 4 vertices, *viz.* θ_{1_min}, θ_{1_max}, θ_{2_min} & θ_{2_max}.

4. Safe Path in Configuration Space: Logistics & Algorithm

4.1. Perspective

Based on the formation of c-space maps, collision-free paths are to be ascertained in 2D as well as in 3D. We will analyse the gamut in four functional *quadrants*, which have been conceptualised from the point of view of (a) disposition of the obstacles (i.e. planar or spatial) and (b)

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

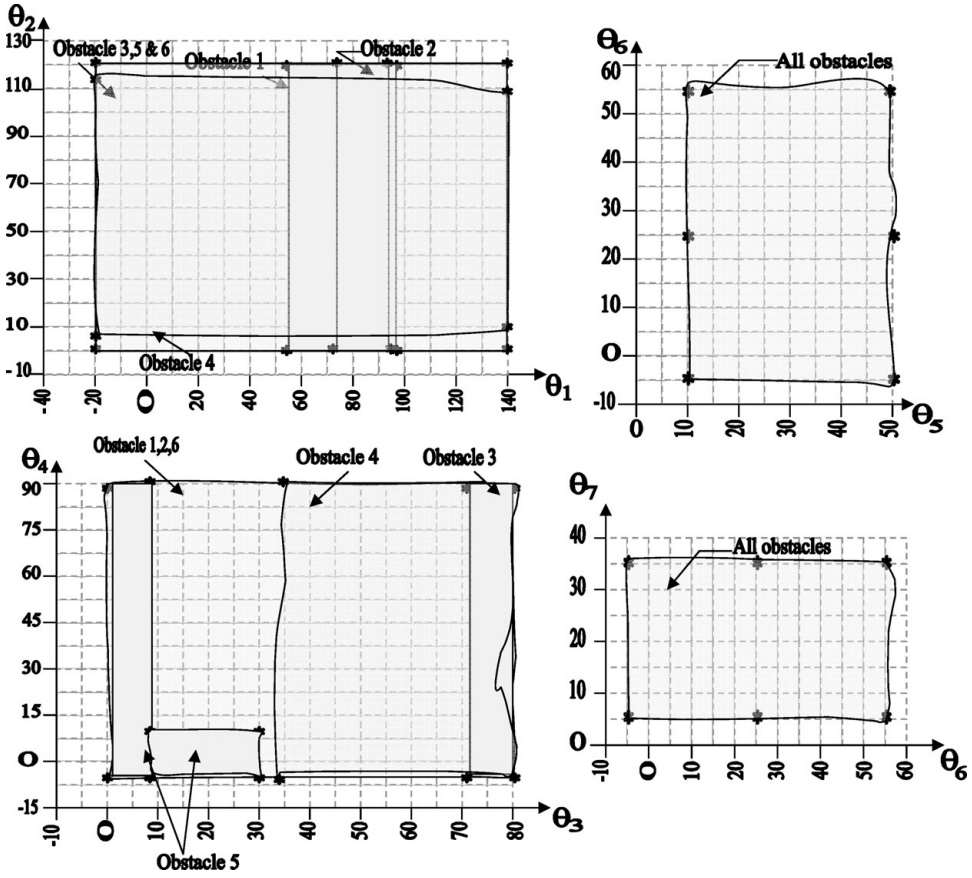


Fig. 18. Four c-space plots for the seven joint revolute robot amidst cluttered workspace of Fig. 17.

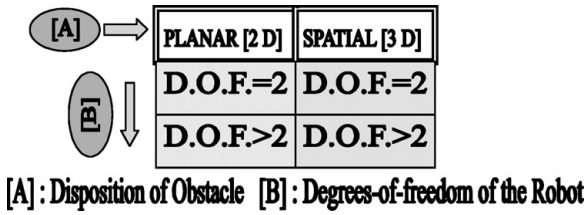


Fig. 19. Schematic of the robotic path planning scenario using Configuration Space approach.

kinematics of the manipulator (i.e. its degrees-of-freedom). Following checker-box illustrates the situation pictorially (refer Fig. 19).

In fact, the methodologies to be adopted for different situations of robot path planning vary significantly and those depend on the task-space nature (i.e. planar or spatial) and the robot kinematics (i.e degrees-of-freedom). Besides, the 3D path planning is also dependent upon the nature of the slices produced from the task space. Those can be identical in nature or non-identical. Table IV presents the scenario, highlighting the methods used for the collision-free path planning.

4.2. Logistics of visibility graph formulation: our model

The workspace of the robot has been modelled by formulating the *visibility matrix* of the v-graph[2] of the cluttered environment. This matrix has been conceived as a kind of 'adjacency' relationship and framed by knowing the necessary visibility information about the nodes of the graph. Figure 21 illustrates the visibility matrix, [Vij], as developed from the environment shown in Fig. 20, which depicts a typical sub-visibility graph, generated out of several

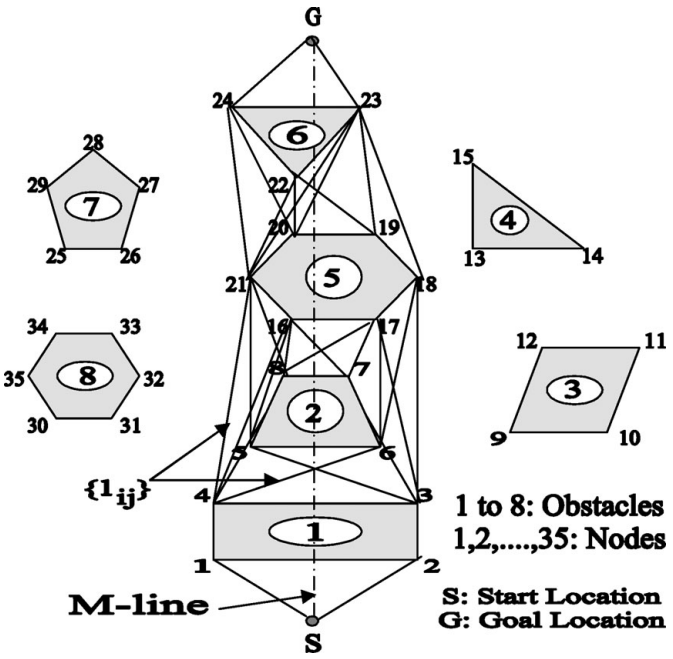


Fig. 20. A representative sub-visibility graph in two dimensions.

[2] In order to reduce computational burden, 'Sub-visibility Graph' technique has been adopted in the present study. It considers only those obstacles in the robot workspace, which collide directly with the M-line.

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

Table IV. Illustrative summary of the methods used for collision-free path planning of robots.

Parameter	Collision-free path planning					
Task space	2 D [Planar]		3 D [Spatial]			
Robot type	D.O.F. = 2	D.O.F. > 2	D.O.F. = 2		D.O.F. > 2	
Method used for path planning			Identical slice	Non-identical slice	Identical slice	Non-identical slice
	(a) c-space Map & (b) V-graph	(a) (Multiple) c-space Maps ⇒ (b) Most Significant c-space Map & (c) V-graph	(a) Sliced c-space Maps ⇒ (b) V-graph based Paths & (c) Final path	(a) Sliced c-space Maps ⇒ (b) V-graph based Paths & (c) *Intersection* of the Feasible paths	(a) (Multiple) c-space Maps ⇒ (b) Most Significant c-space Map & (c) V-graph for each slice & (d) Final path	(a) (Multiple) c-space Maps ⇒ (b) Most Significant c-space Map & (c) V-graph for each slice & (d) *Union* of the Feasible paths

	S	1	2	3	4	5	6	7	8	16	17	18	19	20	21	22	23	24	*G*
S –	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1 –	0	0	2	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2 –	0	0	0	3	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0
3 –	0	0	0	0	4	0	6	7	0	0	17	18	0	0	0	0	0	0	0
4 –	0	0	0	0	0	5	6	0	8	16	0	0	0	0	21	0	0	0	0
5 –	0	0	0	0	0	0	6	0	8	16	0	0	0	0	21	0	0	0	0
6 –	0	0	0	0	0	0	0	7	0	0	17	18	0	0	0	0	0	0	0
7 –	0	0	0	0	0	0	0	0	8	16	17	18	0	0	0	0	0	0	0
8 –	0	0	0	0	0	0	0	0	0	16	17	0	0	0	21	0	0	0	0
16 –	0	0	0	0	0	0	0	0	0	0	17	0	0	0	21	0	0	0	0
17 –	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0
18 –	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	23	0	0
19 –	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	22	23	0	0
20 –	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	22	23	24	0
21 –	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	23	24	0
22 –	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	24	0
23 –	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	*G*
24 –	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	*G*
G –	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 21. The visibility matrix developed from the environment shown in Fig. 20.

polygonal obstacles known *a-priori* (i.e. with pre-fixed locations).

The matrix is of the order $(N + 2) \times (N + 2)$, 'N' being the total number of intermediate nodes of the graph. The full matrix is formed by adding the 'start' (S) and 'goal' (G) nodes, making [Vij] a square matrix. Each row of [Vij] gives the visibility information of that very node. For example, the fifth row of the matrix signifies that the node no. 4 (considering 'S' as the first node) can 'see' nodes 5, 6, 8, 16 & 21 only.

Features of the visibility matrix are listed below:

1. Nodes are numbered in ascending order from 'S' to 'G' maintaining counter-clockwise sense for each obstacle. Backtracking of the nodes is not allowed. Symbolically, if 'n_r' is the particular node under consideration which can see nodes $n_{j1}, n_{j2}, \ldots\ldots\ldots\ldots, n_{jp}, \ldots\ldots, n_{jk}$ then:

2. $$n_{j1} > n_r \quad \text{and} \quad n_{r-1} \leq n_{jp} \leq N \tag{21}$$

3. where 'n_{jp}' is any general node of the graph, *visible* by the node 'n_r' and 'N' is the goal node of the nodal series, namely 1,2,3, ... , *N*.
4. Numbering of nodes is exhaustive and independent of the obstacle nomenclature. In other words, 'n_{jp}' is invariant to obstacle number and also does not have any correlation with any specific obstacle geometry in any form.
5. The entire matrix has got two parts symmetrically disposed off by the diagonal. Because of the reason stated earlier, the generalised element, 'a_{ij}' of [Vij], which signifies the visibility information of the *j*th node as *viewable* with the *i*th node as *viewer*, and can be expressed mathematically as:
6. $$\begin{aligned} a_{ij} &= 0 \text{ or } j, & \forall\, j \geq i \\ a_{ij} &= 0 & \forall\, j \leq i \\ a_{i1} &= 0 & \forall\, i \\ a_{Nj} &= 0 & \forall\, j \end{aligned} \tag{22}$$

7. For simplicity in computation, the original square matrix [Vij] of order $(N + 2)$ has been truncated to a square matrix of order $(N + 1)$, by deleting the first column and the last row. Since all the elements of these row and column are zero as per proposition, this modification will not alter the final result. Obviously, structure of this reduced matrix depends on the modelling of the robot environment and the total number of nodes present.
8. It has been found from the composite evaluations that the total number of computations required for the path planning algorithms is slightly less than $O\ (N^2)$, 'N' being the total number of nodes in the v-graph, corresponding to the workspace under investigation.
9. Essentially the visibility matrix reduces to the following structure in its most practical form, *viz.*:

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

10.

$$[\text{Vij}] = \begin{bmatrix} 0 & 0 & d & d & d & 0 \\ 0 & 0 & 0 & 0 & d & d \\ 0 & 0 & 0 & 0 & d & 0 \\ 0 & 0 & 0 & 0 & 0 & d \\ 0 & 0 & 0 & 0 & 0 & d \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where, 'd' signifies the locations for non-zero entry. The left triangular matrix will be zero in a majority of cases, alongwith the diagonal elements. However, the visibility matrix may include some non-zero entries too corresponding to 'sculptured obstacles' with multiple vertices. With this structure of [Vij], which is time optimised, the nodal lines can be computed as:

$$\{\text{Lij}\} = \sum_{i=1}^{i=N} C_i(a_{ij}^*) \tag{23}$$

where, $\{\text{Lij}\}$ is the nodal line vector, (a_{ij}^*) is the non-zero $[a_{ij}]$ and $C_i(a_{ij}^*)$ is the cardinality of a_{ij}^*. For example, $C_i(a_{ij}^*)$ for the 'S' node (i.e. the first row of [Vij] with $i = 1$) becomes 2.

In addition, the total number of imaginary lines in the v-graph can be computed as:

$$\{\text{Lij}^*\} = \{\text{Lij}\} - \sum_{k=1}^{k=N} d_k \tag{24}$$

where, 'd_k' is the number of edges of the kth· obstacle.

11. A quick estimation of the computational time for the algorithmic path planning using a finalised v-graph of the robot workspace reveals the following:

$$\tau \propto \{\text{Lij}\} \quad \text{or} \quad \{\text{Lij}^*\} \tag{25}$$

where, 'τ' is a factor representing the computational time and the memory requirement factor, say ξ, will be as follows:

$$\xi \propto \{\text{Lij}^*\} \tag{26}$$

Also the range of 'τ' may be estimated for all practical computational situations as,

$$O(N) < \tau \leq O(N^{3/2}) \tag{27}$$

It is to be noted that the lower bound of 'τ' is certainly beyond O (N), while the upper bound can marginally reach $O\,(N^{3/2})$, as 'N' tends to some larger value.

Regarding circular obstacles in 2D plane, a trade-off has been attained between the approximation to the nearest polygonal shape and the computational burden. For example, a perfectly circular-shaped object can be approximated with a circumscribing square-shaped object, but it will be about 80% less efficient in comparison to an approximation with an octagonal shape. Hence, the decision for the optimal selection for approximation for a circular obstacle remains with the overall complexity of the v-graph, i.e. essentially the value of 'N' generated thereby.

4.3. Development of the path planning algorithm in 2D plane

The new heuristic algorithm, namely, *Angular Deviation Algorithm*, has been developed to obtain near-optimal path for the manipulator amidst obstacles in a planar workspace. The formulation of the heuristics and subsequently the solution phase are based on A^* search technique, in general. Following legends have been used in formulating the algorithm.

S: The 'start' location of the robot end effector (in Cartesian or c-space);
G: The 'goal' location of the robot end effector (in Cartesian or c-space);
$L_{M\text{-}Line}$: The imaginary line joining 'S' and 'G';
$L_{M\text{-}Line}$: The geometric length of the imaginary line joining 'S' & 'G', i.e. the 'M-Line';
x: An intermediate level in the graph search process;
V_{x_j}: jth· visible node from xth· node in the v-graph;
$x\ V_{x_j}$: The nodal line, joining the xth· node and the jth· visible node (from xth· node);
i: Iteration number of the graph search process.

This algorithm relies on *angular deviation* as the necessary computing heuristic, which is in-built in nature. It considers each line-segment, $\{l_{ij}\}$, where, $l_{ij} \in \{l\}$, joining the nodes, say, n_i and n_j, where $(n_i, n_j) \in \{n\}$, $\forall\ i, j \in I$, of the sub-visibility graph and computes the angular deviation of that $\{l_{ij}\}$ with respect to the M-line. The logic of this algorithm is to minimise the Angular Heuristic Function, as generated from the angular deviations, at each level of searching. Hence, in a cluttered environment the near-optimal path can be chosen by considering only those line segments, which are comparatively closer to the M-line.

Steps:

1. Initialise the search with $i = 1$.
2. For $i = 1$, loop starts with 'S': note $V_{S_1}, V_{S_2}, \ldots, V_{S_p}$.
3. Compute: $\alpha_{S_1} = \text{Ang (SG, } SV_{S_1})$; $\alpha_{S_2} = \text{Ang (SG, } SV_{S_2}), \ldots\ldots\ldots\ \alpha_{S_p} = \text{Ang (SG, } SV_{S_p}.)$. 4. Check: min. $(\alpha_{S_1}, \alpha_{S_2}, \ldots\ldots\ldots\ldots, \alpha_{S_p})$.
4. If $V_{S_p}. =$ 'G', then stop.
   Else,
5. $i = i + 1$.
6. Begin the next level of search from the xth· node with min. (α_{S_x}).
7. Compute: $\{\alpha_{x_j}\} = \{\text{Ang (SG, } XV_{x_j})\}$, where $x < j \leq N$, 'N' being the total number of nodes in the graph.
8. Go on searching likewise till 'G' occurs and finally note the nodes of the optimal path, so achieved.

4.4. Paradigms of path planning in 3D space for two degrees-of-freedom robots

4.4.1. Model for c-space slices and path generation. The concept of discretisation of robot workspace in preferential 2D slices has been applied for generating safe path in a spatial manifold. As a result, multiple c-space slices are

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

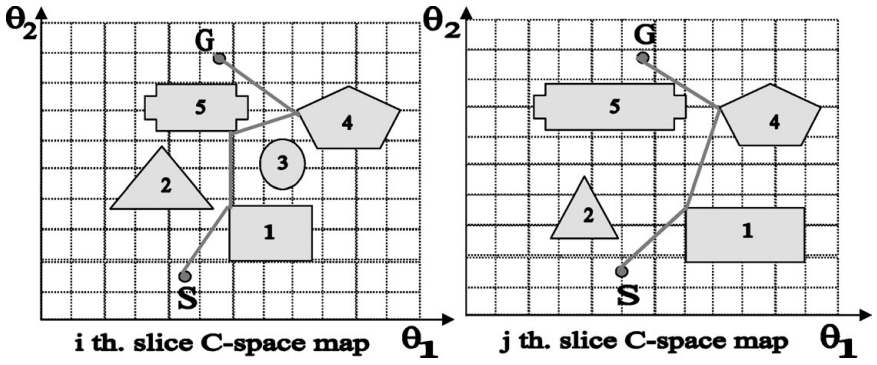


Fig. 22. Schematic view of the sliced c-space maps for a two degrees-of-freedom robot workspace.

generated depending upon the features of the individual slices. Safe paths are then determined, using the *Angular deviation algorithm*, separately for each such c-space slice produced. Thus, if a spatial workspace is segregated in 'k' slices, then there will be 'k' c-space slices also. Figure 22 schematically presents the view of the *sliced c-space maps* for a known environment.

It may be noted that we need to generate *c-space slices*, as shown in Fig. 22, for all the slices equally since our c-space mapping algorithms are in 2D and those are based on planar collision avoidance principles concerning *Point*, *Line* & *Circle* obstacles. Since the total number of slices for a specific environment is fixed *a-priori*, it may so happen that a particular slice of an object is not falling under the obstruction zone of the robot. In that case, that particular slice of that object will not appear in the corresponding c-space slice (e.g. refer *j*th· slice c-space map of Fig. 22, wherein the slice for obstacle 3 is absent). Nonetheless, the axiom followed is *if the last slice of any obstacle is collidable, then all its previous slices ought to be. But the vice-versa is not true*.

Once the requisite c-space slice maps are generated, we need to evaluate the *safe* paths in each of these sliced c-space maps using the visibility-based Angular deviation algorithm. Thus a set of paths will be obtained and the cardinality of the set will be equal to the number of slices. Finally we will take the *intersection* of the possible paths, as only intersection set will represent the optimal path in true sense. However, intersection won't be the solution for situations where slice(s) for obstacle(s) is/are absent in a particular c-space slice. For example, with reference to Fig. 22, considering two c-space slices in total, we have to follow the path shown in the first map, i.e. the *i*th· slice map.

On the other hand, in situations where all the object-slices are present in all the c-space slices, then intersection can be advantageous, as we can omit longer routes via nodes in certain instances. A typical case is exemplified in Fig. 23. Here a particular object is shown to have slightly different geometries and the path between 'S' & 'G' also varies accordingly as shown. In case (a), we are unable to consider $2'$ as 'node', because of the proposition of v-graph and the path (*viz.* $S \Rightarrow \ldots\ldots 1 \Rightarrow 2 \Rightarrow 3 \Rightarrow$ G) is bound to pass through the stipulated node 2 only. However, node $2'$ lies very much inside the boundary of the c-space obstacle and in fact, it is closer to 'G' as compared to node 2. Thus the path shown in case (b) is the optimal solution (*viz.* $S \Rightarrow \ldots\ldots\ldots 1 \Rightarrow 2' \Rightarrow G$) and in fact, it is also the intersection of the two alternatives.

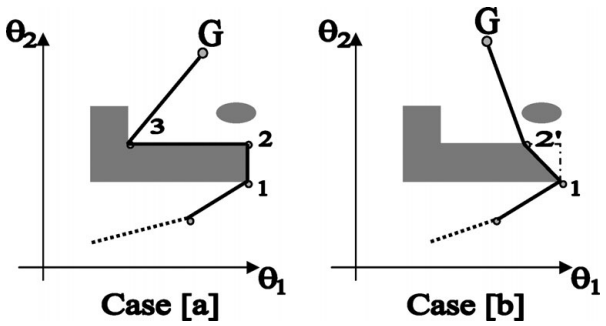


Fig. 23. Selection of path using intersection of alternatives (paths).

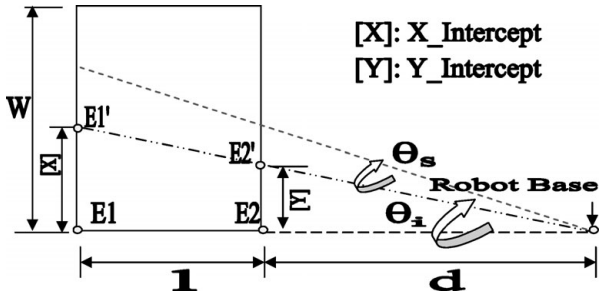


Fig. 24. Schematic representation of the dimensional metrics of a 2D 'slice'.

4.4.2. Quantitative evaluation of c-space slice points. In order to evaluate the c-space points mathematically, the relevant algorithm generates the intercept co-ordinates (*X* & *Y*) corresponding to each 'slice', which are governed by the rotational range of the robot waist and the finite resolution of the waist rotation. The program is applicable only to rectangular 3D solid obstacles, e.g. cube, rectangular parallelopiped, pyramid etc., either directly or after duly transformed from spherical or semi-spherical obstacles. The model is being illustrated schematically through Fig. 24.

The relevant formulation vis-à-vis algorithm of the concerned model is described in detail below. Consider Fig. 24, let:

w : Width of the obstacle;
d : Distance between the robot and the obstacle;
(x_b, y_b) : Co-ordinates of the robot base;
(x_1, y_1) & (x_2, y_2) : Co-ordinates of the edge of the obstacle in consideration in 2D elevation;

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

$\theta_{b_max.}$ & $\theta_{b_min.}$: Maximum and minimum values of the base rotational edge;
θ_S: Slicing value of the base rotational angle.

The range of base rotation is computed as,

$$r_{\text{base}} = (\theta_{b_\max.} - q_{b_\min.}) \tag{28}$$

Length of the 'edge' of the obstacle is,

$$l = [(x_2 - x_1)^2 + (y_2 - y_1)^2]^{1/2} \tag{29}$$

Now, only one slice is possible, if,

$$d(\pi\ \theta_S/180) > w \tag{30}$$

In situations where more than one slice is possible for a particular obstacle, two cases can appear.

Case I: Robot base is in-line with the obstacle
In this case, let $S = d\ (\pi\ \theta_i/180)$, where $\theta_S \leq \theta_i \leq r_{\text{base}}$.
If $S \leq w$, then 'slice' is possible and co-ordinates of the intercept of the slices are given by,

$$x_{\text{int_1}} = x_1 + [l\ \tan\ \theta_i/(l/d)]\,[1 + (l/d)] \tag{31}$$

$$y_{\text{int_1}} = y_1 + [l\ \tan\ \theta_i/(l/d)]\,[1 + (l/d)] \tag{32}$$

$$x_{\text{int_2}} = x_2 + [l\ \tan\ \theta_i/(l/d)] \tag{33}$$

$$y_{\text{int_2}} = y_2 + [l\ \tan\ \theta_i/(l/d)] \tag{34}$$

where, $(x_{\text{int_1}},\ y_{\text{int_1}})$ and $(x_{\text{int_2}},\ y_{\text{int_2}})$ are one set of co-ordinates corresponding to one slice. With θ_i varying within its range with an increment of θ_S, the other set of slice co-ordinates will be obtained.

Case II: Robot base is not in-line with the obstacle
In this case,

$$\theta_{in} = \tan^{-1}[|(y_2 - y_b)/(x_2 - x_b)|] \tag{35}$$

Also,

$$S' = d[\pi(\theta_j - \theta_{in})/180],\ \forall\ j,\ \text{where}\ \theta_{in} \leq \theta_j \leq r_{\text{base}}. \tag{36}$$

If $S' \neq 0$ and $S' \leq w$, then slice is possible and co-ordinates of the intercept of the slices are given by,

$$x_{\text{int_1}n} = x_1 + [l\ \tan(\theta_j - \theta_{in})/(l/d)]\,[1 + (l/d)] \tag{37}$$

$$y_{\text{int_1}n} = y_1 + [l\ \tan\,(\theta_j - \theta_{in})/(l/d)]\,[1 + (l/d)] \tag{38}$$

$$x_{\text{int_2}n} = x_2 + [l\ \tan\,(\theta_j - \theta_{in})/(l/d)] \tag{39}$$

$$y_{\text{int_2}n} = y_2 + [l\ \tan(\theta_j - \theta_{in})/(l/d)] \tag{40}$$

As before, $(x_{\text{int_1}n}, y_{\text{int_1}n})$ and $(x_{\text{int_2}n}, y_{\text{int_2}n})$ are one set of co-ordinates corresponding to one slice. With θ_j varying within its range with an increment of θ_S, the other set of slice co-ordinates will be obtained. Selection of 'safe' nodes of the robot end effector in the 3D space depends on the elemental results, as obtained from the corresponding planar analysis. Symbolically, if the collision-free path for one particular 'slice' is represented as,

$$\{P_{S_k}\} = \{S, X_1, X_2, \ldots\ldots\ldots\ldots, G\}_{S_k} \tag{41}$$

where, $X_1, X_2, \ldots\ldots\ldots\ldots\ldots$, represent the serial number of the 'safe' nodes (may not be in the same order as the node nos., *viz.*, 1,2,3,.....) and 's_k' is the *k*th· slice, then the final path will be the union of all such feasible combinations, *viz.*,

$$\{P\} = \bigcup_{k=1}^{k=n} \{P_{S_k}\} \tag{42}$$

where, 'n' is the total number of slices generated.

4.5. Evaluation of path in 3D for higher dimensional robot
As depicted in Fig. 19, evaluation of the collision-free path in 3D will depend on the degrees-of-freedom of the robot (i.e. whether d.o.f. = 2 or >2). Nonetheless, in both the cases we need to fragment the 3D task space in multiple slices, which may or may not be identical to one another. Thus, we will arrive at a situation wherein we have to deploy different strategies to evaluate a safe path. These models are described below.

4.5.1. Model for obtaining safe path for identical slices. The first and foremost pre-requisite of evaluating a collision-free path in this case is to have the *most significant 2D c-space slice map* for the robotic workspace under consideration, as described in 3.4.2. Once the critical c-space slice map is earmarked, the next task is to pin-point the corresponding locations for 'S' & 'G' in that map. This can be achieved by using the inverse kinematic solution for 'S' & 'G' and subsequent mapping from task space to joint space. For example, consider again the case of a seven degrees-of-freedom robot shown in Fig. 9, wherein say the $[\theta_3$–$\theta_4]$ map is the most significant. Thus, the corresponding v-graph of the environment will have non-identical 'S' & 'G' signatured as, *S*: $(\theta_3 = \alpha^\circ, \theta_4 = \beta^\circ)$ and *G*: $(\theta_3 = \gamma^\circ, \theta_4 = \delta^\circ)$. We will assume that the set of other joint angles, i.e. $\{\theta_1, \theta_2, \theta_5, \theta_6, \theta_7\}$ to be constant throughout the process of path generation. Now, by using the developed path planning algorithm, we will finally get a collision-free optimal path between *S*: (α, β) and *G*: (γ, δ). The generalised representation of co-ordinates of any two nodes (say N_i & N_j) of the said path will be as follows,

$$N_i \equiv \{(\theta_1 = c_1,\ \theta_2 = c_2), \theta_3 = x_i, \quad \theta_4 = y_i,$$
$$(\theta_5 = c_5, \theta_6 = c_6, \theta_7 = c_7)\}\ \text{and}$$
$$N_j \equiv \{(\theta_1 = c_1, \theta_2 = c_2), \theta_3 = x_j, \theta_4 = y_j,$$
$$(\theta_5 = c_5, \theta_6 = c_6, \theta_7 = c_7)\}$$

Where $[c_1$ & $c_2]$ and $[c_5,\ c_6$ & $c_7]$ are two non-identical group of constants to be evaluated using inverse kinematics solution for 'S' and 'G' respectively.

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

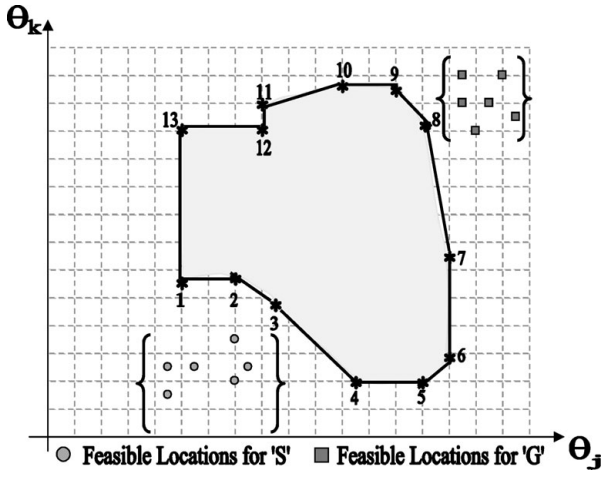


Fig. 25. Multiple feasible locations for 'S' & 'G' inside a specific c-space plot.

The general lemma in this regard is stated as below,

$$N_i \equiv \{(\theta_1 = c_1, \theta_2 = c_2, \ldots\ldots, \theta_{p-1} = c_{p-1}), \theta_p = x_i, \theta_q = y_i, (\theta_{q+1} = c_{q+1}, \theta_{q+2} = c_{q+2,\ldots}, \theta_k = c_k)\} \text{ and}$$

$$N_j \equiv \{(\theta_1 = c_1, \theta_2 = c_2, \ldots\ldots, \theta_{p-1} = c_{p-1}), \theta_p = x_j, \theta_q = y_j, (\theta_{q+1} = c_{q+1}, \theta_{q+2} = c_{q+2,\ldots}, \theta_k = c_k)\}$$

where, k: the degrees-of-freedom of the articulated robot; θ_1, $\theta_2, \ldots\ldots \theta_p, \theta_q, \ldots\ldots, \theta_k$: joint angles of the robot of which 'p' & 'q' represent any two consequtive pair; $[\theta_p$–$\theta_q]$: the most significant c-space slice map; $[c_1, c_2, \ldots\ldots, c_{p-1}]$ & $[c_{q+1}, c_{q+2}, \ldots, c_k]$: two non-identical group of constants to be evaluated using inverse kinematics solution for 'S' and 'G' respectively.

Once this path is obtained for a particular slice, say the first one, the same procedure can be repeated for other slices too, because all the slices are identical. And, obviously the same path will be obtained in all slices, which will be designated as the final path for that robot in 3D. The kinematic inversion for 'S' & 'G' is another important facet in this regard and analytically, there can be multiple feasible positions of 'S' as well as 'G' in the c-space plot (refer Fig. 25). One each from the two clusters of feasible locations can be selected for S: (α, β) and G: (γ, δ).

4.5.2. Model for obtaining safe path for non-identical slices. In this case, we will get different sets of nodal points, corresponding to safe paths, for each slice. The procedure for obtaining a particular set of nodal points (nodes) for a specific slice is same as described in 4.5.1. But the challenge involved here is the most significant c-space slice is not fixed for the slices, rather in worst case it will differ. For example, let us take the case of seven d.o.f. manipulator and we assume there are five slices in the workspace. After c-space slice mapping, we find that while $[\theta_3$–$\theta_4]$ is the most significant map for the first slice, $[\theta_1$–$\theta_2]$ is the same for second slice. And likewise, the maps of $[\theta_3$–$\theta_4]$, $[\theta_5$–$\theta_6]$ & $[\theta_1$–$\theta_2]$ are the significant ones for third, fourth and fifth slice respectively. Thus the hurdle becomes in unifying these varying sets of maps into one final path. This is solved considering the *union* of the available sets of nodal points.

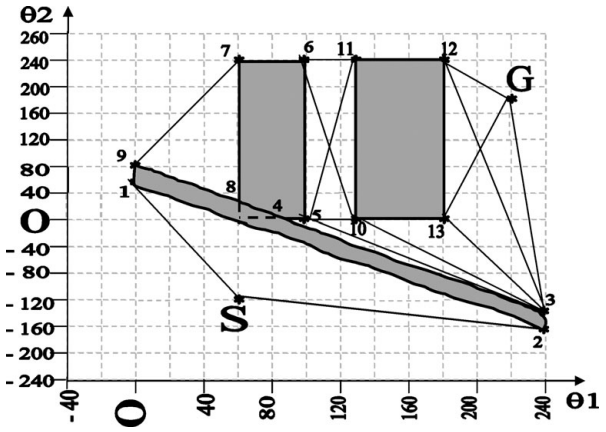


Fig. 26. Visibility graph generated out of the 2D environment, vide Fig. 6.

In general, if there are 'k' non-identical slices and the sets of nodal points for each safe path are represented by, $\{S_k\} = \{N_{1k}, N_{2k}, \ldots\ldots\ldots, N_{qk}\}$, where 'q' is the cardinality of the set and the value of 'q' may vary for different 'k', then the final path will be defined as,

$$\{S_{\text{Final}}\} = \bigcup_{p=1}^{p=k} \{S_p\}.$$

In other words, statistical union of nodes will proceed slice-wise; i.e. to get the *safe* path complete posting all the nodes under one slice, then move on to the next slice and so on, till all the slices are exhausted. Nonetheless, the co-ordinates of the nodes in the path will be evaluated as per the lemma described in 4.5.1.

4.6. Illustrative examples

The developed path planning algorithm has been tested with two sample environments, the first one is contains a two degrees-of-freedom robot while the second one includes a seven degrees-of-freedom robot.

4.6.1. Sample workspace for two degrees-of-freedom robot. This example has a reference to the robot workspace with circular obstacles, as shown in Fig. 6 and subsequently, the c-space map, vide Fig. 8. Figure 26 presents the final c-space obstacles[3] with nodes numbered sequentially and the v-graph generated there from.

Table V shows the output of the graph search process using our algorithm, developed with 'S' & 'G' configurations as $(60^\circ, -120^\circ)$ and $(222^\circ, 190^\circ)$ respectively. For comparison, the result obtained through A* search algorithm is also included.

It may be mentioned here that we can very well *benchmark* the result obtained by the Angular Deviation Algorithm, as it is representative amongst the *AI-based* heuristic algorithms.

[3] Only obstacle 3, 5 & 6, referred in Figs. 6 and 8, have been considered for the creation of visibility graph in order to reduce computational complexity.

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

Table V. Collision-free near-optimal path between 'S' & 'G' with reference to example in Fig. 26.

Algorithms used	Path with nodes	Joint angle combination (in degrees)
Angular deviation algorithm	$S \Rightarrow 2 \Rightarrow 3 \Rightarrow G$	(60, −120); (240, −166); (240, −139); (222, 190)
A* Algorithm	$S \Rightarrow 1 \Rightarrow 9 \Rightarrow 7 \Rightarrow 13 \Rightarrow 12 \Rightarrow G$	(60, −120); (0, 66); (0, 86); (67, 240); (129, 240); (187, 240); (222, 190)

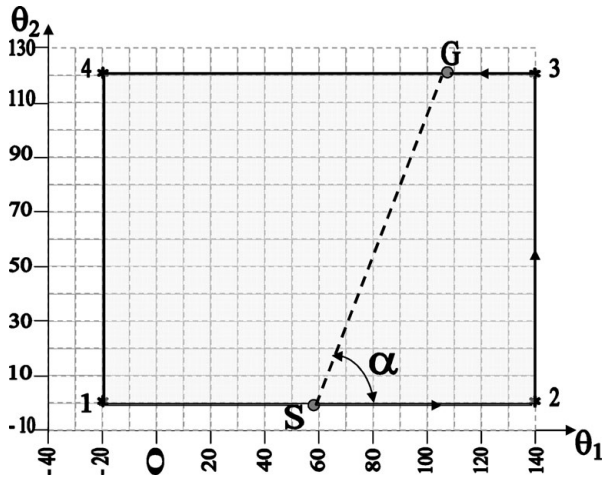


Fig. 27. Visibility graph obtained as per the most significant c-space slice plot of Fig. 18.

In fact, due to its logistics, the developed algorithm is having an edge over the other possible metrics of path planning, e.g. Generalised Voronoi Diagram (GVD), Cellular Automata and Potential Field. All of these three methods rely on diversification in search, which eventually leads to more computational time and complexity. Besides, the important attribute, namely, '*closeness to desired path*' is compromised in most of the non-AI based techniques. In comparison, AI-based searches are much robust and coherent; like the case with Angular Deviation Algorithm. The other group of search algorithms, based on mathematical programming, such as Variational Methods, Hierarchical Dynamic Programming and Tangent Graph Method, are although relatively better focused, but those are highly *memory-extensive*. Thus, in all counts, Angular Deviation Algorithm scores high amongst the various alternatives in graph-search methods.

4.6.2. Sample workspace with seven degrees-of-freedom robot. This example is in reference to the robotic environment shown in Fig. 17 and subsequently the various c-space slice maps, as detailed in Fig. 18. As we have declared in Section 3.5 that $[\theta_1$–$\theta_2]$ plot is the most significant out of the four plots, we need to obtain the v-graph for this plot. Figure 27 shows the developed v-graph for this c-space slice plot. Here the generated v-graph is relatively simpler by default as it has only four nodes, which is due to the fact that both the joint angles in consideration, *viz.* θ_1 & θ_2 are plotted in their full ranges. Thus 'S' & 'G' are also located on the boundary lines, because other locations will be infeasible.

However, it is to be noted that the exact shape of the v-graph (generated out of the most significant c-space slice plot) will depend upon the joint angle ranges of the joint pair under consideration and we will have distinct locations for 'S' & 'G', outside the c-space zone. It is evident from Fig. 27 that $S \Rightarrow 2 \Rightarrow 3 \Rightarrow G$ is the optimal path as 'α' is the smaller angle, which guides this path as per Angular deviation algorithm.

The generalised formulation for evaluating the angular position of 'S' & 'G' in the v-graph (using inverse kinematics routine for the manipulator) is as follows,

$$\sum_{j=1}^{j=7} l_j \cos\left(\sum_{k=1}^{k=7} \theta_k^m\right) = X_m \tag{43a}$$

$$\sum_{j=1}^{j=7} l_j \sin\left(\sum_{k=1}^{k=j} \theta_k^m\right) = Y_m \tag{43b}$$

where. 'm' : positional attribute of the end-point, i.e. either 'S' or 'G'; $\{l_j\}$: link lengths; $\{\theta_k^m\}$: joint angles for 'S' or 'G' and (X_m, Y_m): planar Cartesian co-ordinates for 'S' or 'G'.

Now considering the Cartesian co-ordinates for 'S' as (20, 72.5) and the constant values for $\{\theta_3\ \theta_4\ \theta_5\ \theta_6\ \theta_7\}$ as [10° 5° 15° 6° 10°] we can solve for θ_1 & θ_2 using eqns. (43a,b), which gives us $\theta_1^S \cong 60°$ and $\theta_2^S \cong 0°$. Similarly considering 'G' as (−30, −40) with the constant values for $\{\theta_3\ \theta_4\ \theta_5\ \theta_6\ \theta_7\}$ as [5° 8° 18° 4° 13°] we can solve for θ_1 & θ_2, which gives us ${\theta_1}^G \cong 108°$ and $\theta_2^G \cong 120°$. Thus, as proposed in Section 4.5.1, the nodes, $\{N_k, \forall k = 1,2,3,4\}$ of the final collision-free path of the manipulator between 'S' & 'G' will be: $N_1 \equiv$ 'S' = (60°, 0°, 5°, 8°, 18°, 4°, 13°); $N_2 = (140°, 0°, 5°, 8°, 18°, 4°, 13°)$; $N_3 = (140°, 120°, 5°, 8°, 18°, 4°, 13°)$ and $N_4 \equiv$ 'G' = (108°, 120°, 5°, 8°, 18°, 4°, 13°).

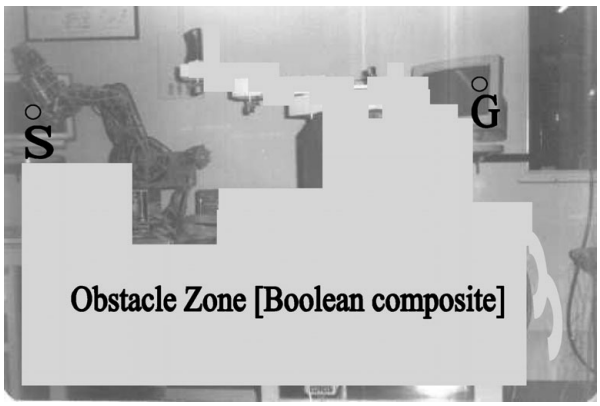


Fig. 28. Photographic view of the test set-up for spatial path planning with RHINO robot.

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

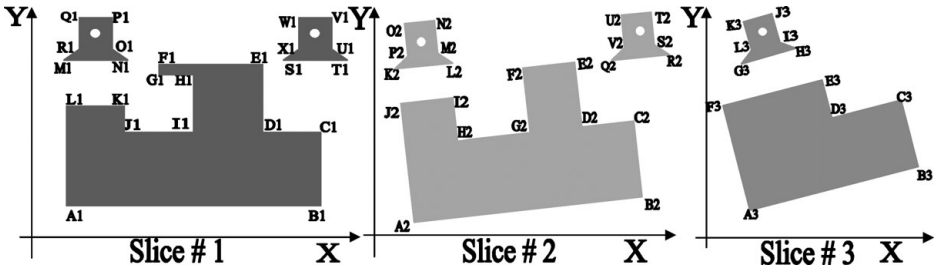


Fig. 29. Schematics of the Cartesian task-space slices for the case-study with RHINO robot.

5. Case Study

We have studied one real-life case of robot path planning in 3D, based on c-space modelling and v-graph searching, as delineated in the paper so far. The study was made with a five degrees-of-freedom articulated robot, RHINO-XR 3®, during its traverse between two pre-defined spatial locations through a collision-free path. The main focus was to maneuver this robot between 3D obstacles in reaching a goal location in a cluttered (laboratory) environment. Since RHINO is a low-payload robot, instead of standard pick-and-place tests, we designed our experiment such that it had to only *touch* the start ('S') and goal ('G') locations by the gripper end-point. The safe path in 3D, between the start and goal locations, was arrived using *c-space slice* mapping and *Angular Deviation Algorithm* (refer Section 4.3). Figure 28 presents the photographic view of the experimental set-up, emphasising the combined obstacle zone.

Based on the obstacle zone map vis-à-vis waist rotation of the RHINO robot, we have discretised the workspace into three *non-identical* slices. The task spaces, corresponding to these slices, are schematically shown in Fig. 29. In all the sliced maps, the vertices of the combined obstacles are labeled alphabetically, with a numeric indication for the slice-number. For example, the vertex 'A1' signifies the vertex number 'A' in slice number1. It is to be noted that the vertex numbers are not *obstacle-specific*, rather those are serially numbered depending on the shape of the obstacle-zone in that very slice.

The co-ordinates of the vertices under each of the three slices were obtained by physical measurement of the task space in 3D. In other words, first we took the measurements of the Boolean obstacle(s) in (x, y, z) form and then the planar co-ordinates of the slice-vertices were evaluated using the method of *projection geometry*. The co-ordinates of the vertices, so evaluated, under each slice, are tabulated in the matrix below.

$Slice\ 1 \Rightarrow$	$\{A1 : (10, 20)$	$B1 : (190, 20)$
$C1 : (190, 68)$	$D1 : (145, 68)$	$E1 : (145, 98)$
$F1 : (85, 98)$	$G1 : (85, 80)$	$H1 : (95, 80)$
$I1 : (95, 68)$	$J1 : (45, 68)$	$K1 : (45, 78)$
$L1 : (10, 78)$	$M1 : (15, 128)$	$N1 : (45, 128)$
$O1 : (40, 135)$	$P1 : (40, 155)$	$Q1 : (20, 155)$
$R1 : (20, 135)$	$S1 : (210, 128)$	$T1 : (240, 128)$
$U1 : (235, 135)$	$V1 : (235, 155)$	$W1 : (215, 155)$
$X1 : (215, 135)\}$		

$Slice\ 2 \Rightarrow$	$\{A2 : (15, 25)$	$B2 : (195, 35)$
$C2 : (192, 54)$	$D2 : (170, 58)$	$E2 : (160, 88)$
$F2 : (142, 68)$	$G2 : (140, 58)$	$I2 : (45, 68)$
$J2 : (15, 68)$	$K2 : (18, 130)$	$L2 : (52, 132)$
$M2 : (47, 138)$	$N2 : (48, 142)$	$O2 : (22, 150)$
$P2 : (22, 132)$	$Q2 : (212, 138)$	$R2 : (242, 140)$
$S2 : (236, 136)$	$T2 : (236, 156)$	$U2 : (222, 154)$
$V2 : (220, 134)\}$		

$Slice\ 3 \Rightarrow$	$\{A3 : (18, 28)$	$B3 : (198, 38)$
$C3 : (193, 56)$	$D3 : (172, 58)$	$E3 : (168, 88)$
$F3 : (142, 88)$	$G3 : (18, 134)$	$H3 : (52, 134)$
$I3 : (46, 138)$	$J3 : (48, 144)$	$K3 : (24, 148)$
$L3 : (24, 128)\}$		

The co-ordinates of the 'S' and 'G' are measured prior to the experiment and those are (30, 145, 40) & (225, 145, 42) respectively. It may be mentioned that 'S' & 'G' will not appear in the sliced task space(s); rather, those will be only in 3D task space as well as in c-spaces (sliced). Now, considering the kinematics of the RHINO robot, we get a feasible set of joint angle combinations for 'S' & 'G' though inverse kinematics as, 'S': $\{\theta_1 = 50^\circ, \theta_2 = -10^\circ, \theta_3 = 15^\circ, \theta_4 = 50^\circ, \theta_5 = 15^\circ\}$and 'G': $\{\theta_1 = 60^\circ, \theta_2 = 120^\circ, \theta_3 = 58^\circ, \theta_4 = 98^\circ, \theta_5 = 20^\circ\}$.

As evident by now, we will have three non-identical c-space slice maps for this environment and for the clarity in comparison between these three maps, we have selected common scale for the joint angles. For example, the scale of 'θ_1' in *slice* 1 map will be same as that of in *slices* 2 & 3 and likewise, for other joint angles. This universality in scaling is helpful in judging the *most critical* map of a particular slice. We will now present the details of the three c-space slice maps, in their final form, alongwith the demarcation of the most critical map(s).

Figures 30–32 illustrate the conjugate maps corresponding to slice# 1, slice# 2 & slice# 3 respectively.

It is to be noted that some of the c-space slice maps in the above figures bear similarity; in fact, those maps are bounded by rectangular regions, occupying the full rotational ranges[4] of the participating joint angles. That means, the full region is formidable, so far as the selection of safe nodes are concerned. This property is unique in

[4] The *effective* rotational ranges of the five joint-angles of the RHINO robot are, θ_1: (-40° to 120°), θ_2: (-10° to 120°), θ_3: (10° to 90°), θ_4: (10° to 100°) and θ_5: (5° to 65°), as selected on the basis of our task-space layout & experiments.

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

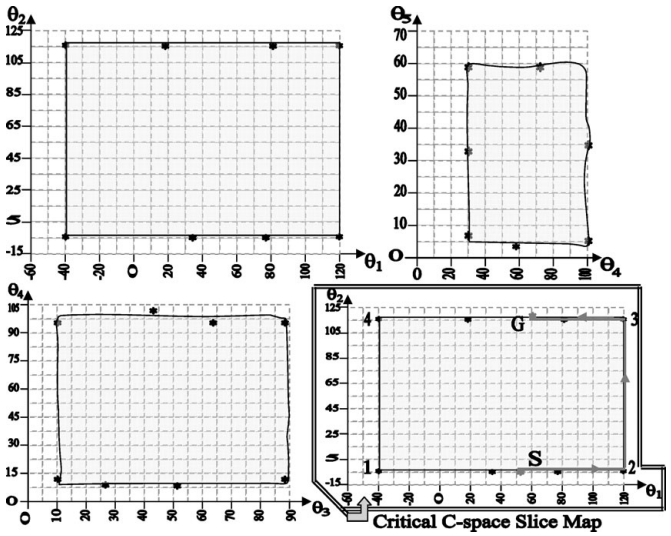


Fig. 30. c-space map for the first slice pertaining to the case study.

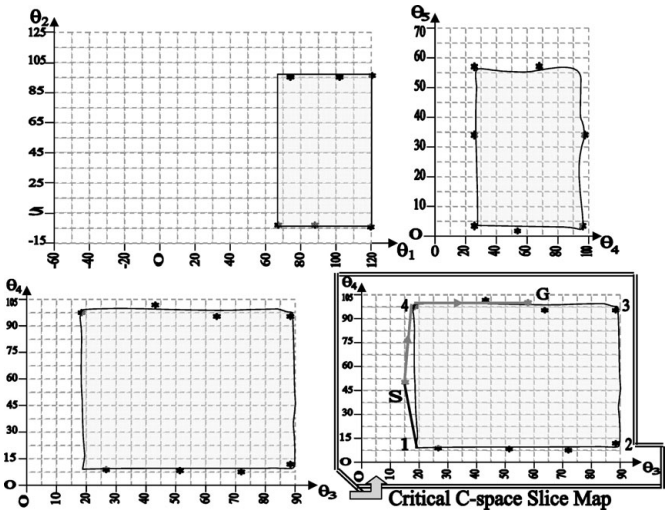


Fig. 31. c-space map for the second slice pertaining to the case study.

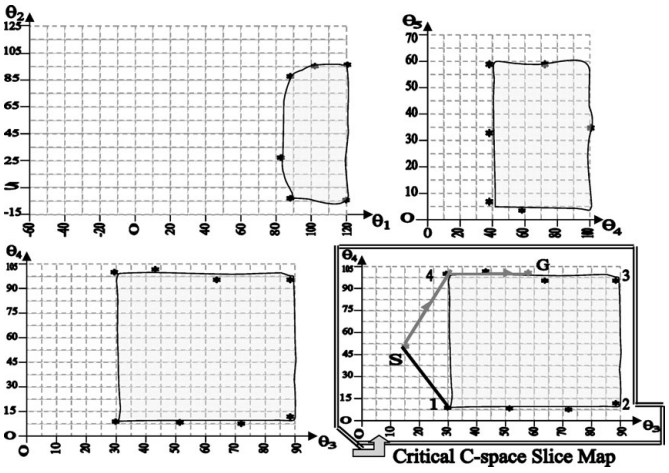


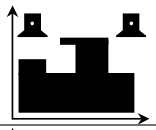
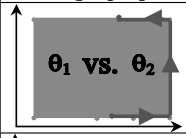
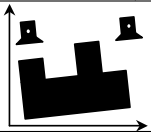
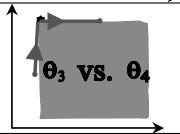
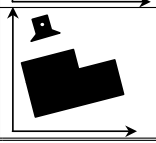
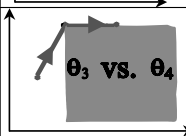
Fig. 32. c-space map for the third slice pertaining to the case study.

the developed method, and it is helpful for obtaining the safe path in the final *go*. Now, assimilating all the three critical c-space slice maps as per Figs. 30–32, we get the final safe path for the environment as the statistical *union* of the slice maps and it is represented as, $S \Rightarrow$ '2'$_{[1]} \Rightarrow$ '3'$_{[1]} \Rightarrow$ '4'$_{[2]} \Rightarrow$ '4'$_{[3]} \Rightarrow G$, where the legend 'N'$_{[k]}$ symbolises node 'N' of the kth slice (refer Section 4.5.2 for the formulation). Thus, the final path has got four *Intermediate Points* (IP), besides 'S' & 'G'. The joint angle combinations for these 'IP$_x$', $\forall x = 1, \ldots, 4$ (as labeled from 'S' onwards) are evaluated as, IP$_1 \equiv$ '2'$_{[1]} \Rightarrow \{\theta_1 = 120^\circ, \theta_2 = -10^\circ, \theta_3 = 58^\circ, \theta_4 = 98^\circ, \theta_5 = 20^\circ\}$; IP$_2 \equiv$'3'$_{[1]} \Rightarrow \{\theta_1 = 120^\circ, \theta_2 = 120^\circ, \theta_3 = 58^\circ, \theta_4 = 98^\circ, \theta_5 = 20^\circ\}$; IP$_3 \equiv$ '4'$_{[2]} \Rightarrow \{\theta_1 = 50^\circ, \theta_2 = -10^\circ, \theta_3 = 20^\circ, \theta_4 = 97^\circ, \theta_5 = 20^\circ\}$ and IP$_4 \equiv$ '4'$_{[3]} \Rightarrow \{\theta_1 = 50^\circ, \theta_2 = -10^\circ, \theta_3 = 30^\circ, \theta_4 = 97^\circ, \theta_5 = 20^\circ\}$.

Table VI presents a summary of the various important outputs pertaining to the case study, with details of the computational time (for PC-based evaluation). Here, *Elapsed Time* has been divided into elemental time-periods (computational) against 4 sub-heads, *viz.* 'A': *Generation of slices in task space with co-ordinates (x, y) & node numbering*; 'B': *Generation of c-space maps, including the critical-most*; '**C**': *Development of the v-graph* and 'D': *Graph searching & output of the Angular Deviation Algorithm*.

It may be noted that elemental timings for module *A*, *B*, *C* & *D* against individual slices give an apprehension regarding

Table VI. Summary data for the case study with details of the computational time [PC-based].

Slice	Task Space	Critical C-space & V-graph path	Elapsed Time (for computation) [sec.]				
			A	B	C	D	Total
1			4.5869	32.3878	16.8239	3.7116	57.5102
2			4.3758	30.9764	16.7132	3.7208	55.7862
3			3.8916	30.3358	16.4581	3.7211	54.4066
Final (Combined Computation) ⇒			**4.7938**	**32.8832**	**16.8423**	**3.7428**	**58.2621**

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

the relative toughness of the corresponding task space and later on c-space & v-graphing. On the contrary, the final combined timings indicate the actual processing time (using multi-session processing of the operating system of the PC) of the problem, with usual co-processor actuations. Similar timings were observed while using A^* *algorithm* for graph search.

6. Conclusions

The details of the visibility graph-based heuristic algorithm for *safe* path planning in 2D plane as well as 3D space have been discussed in the paper, backed up by the theoretical paradigms of the generation of c-space obstacles from their respective task spaces. The outcome of the c-space and v-graph algorithms have been found effective in programming the robot in order to perform certain pre-specified tasks or a series of tasks, such as in somewhat off-the-track industrial applications. The *best* path needs to be selected out of the possible alternatives by considering the most feasible criteria, which is essentially application specific. The novelty of the developed method lies with the ease of computational burden as 2D c-space slices are being joined statistically (*union*). Also by not incorporating all obstacles in one c-space slice we are improving upon computational efficiency and thereby reducing undue technical details regarding the obstacles. However, the safe path obtained by the developed method may overrule some nearer nodes, because the corresponding c-space slices are based on maximum *safety margins*, as per the propositions of the model. In fact, the concept of *formidable zones* is introduced in our model to avert potentially dangerous joint angle configurations and thus, at times, the entire joint angle range-space gets selected for the c-space map. The reason for taking this lemma is to safeguard the robot's motion between 'S' & 'G' to the best extent. Thus we may end up in some joint angle (nodal) combinations, which might have been omitted, but it is always better to select a safe & secured path, rather than risking the robot motion for potential grazing and/or full collision with the obstacle(s). As per the proposed method, c-space slices often look trivial (e.g. regular geometrical shaped obstacles), although those are quite computationally intensive. Nonetheless, the geometrical simplification in appearance makes the v-graph map easy and subsequently the graph-search process too.

Acknowledgments

Gratitude is due to the faculties of the robotics laboratory, department of Production Engineering, Jadavpur University, Kolkata, India for supporting with the case study. The author acknowledges useful contribution made by Shri Sovan Biswas, Infosys Ltd., India in coding the path planning algorithm used in the case study.

References

1. E. U. Acar, H. Chosel, Y. Zhang and M. Schervish, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *Int. J. Rob. Res.* **22**(7–8), 441–466 (2003).
2. C. Bajaj and M. S. Kim, "Generation of configuration space obstacles: Moving algebraic surfaces," *Int. J. Rob. Res.* **9**(1), 92–112 (1990).
3. M. S. Branicky and W. S. Newman, 'Rapid Computation of Configuration Space Obstacles," *Proceedings of the IEEE International Conference on Robotics and Automation*, Cincinnati, OH, USA (1990) pp. 304–310.
4. R. A. Brooks, "Solving the find-path problem by good representation of free space," *IEEE Trans. Syst. Man Cybern.* **SMC-13**(3), 190–197 (1983).
5. R. C. Brost, 'Computing Metric and Topological Properties of Configuration Space Obstacles," *Proceedings: IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, USA (1989) pp. 170–176.
6. D. Campbell and J. Higgins, "Minimal visibility graphs," *Inf. Process. Lett.* **37**(1), 49–53 (1991).
7. B. Curto and V. Moreno, "Mathematical Formalism for the Fast Evaluation of the Configuration Space," *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA (Jul. 1997) pp. 194–199.
8. M. T. De Pedro and R. G. Rosa, "Robot path planning in the configuration space with automatic obstacle transformation," *Cybernatics Syst.* **23**(4), 367–378 (1992).
9. M. Erdmann, "On a representation of friction in configuration space," *Int. J. Rob. Res.* **13**(3), 240–271 (1994).
10. L.-C. Fu and D.-Y. Liu, "An efficient algorithm for finding a collision-free path among polyhedral obstacles," *J. Rob. Syst.* **7**(1), 129–137 (1990).
11. E. G. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE Trans. Rob. Autom.* **RA-1**(1), 21–30 (1985).
12. T. Hasegawa and H. Terasaki, "Collision avoidance: Divide – and – conquer approach by space characterization and intermediate goals," *IEEE Trans. Syst. Man Cybern.* **SMC-18**(3), 337–347 (1988).
13. Y. K. Hwang and N. Ahuja, "Gross motion planning – a survey," *ACM Comput. Surv.* **24**(3), 219–291 (1992).
14. S. Jun and K. G. Shin, "A Probabilistic Approach to Collision-Free Robot Path Planning," *Proceedings of the IEEE International Conference on Robotics and Automation*, Philadelphia, PH, USA (1988) pp. 220–225.
15. S. S. Keerthi and J. Selvaraj, "A fast method of collision avoidance for an articulated two link planar robot using distance functions," *J. Ins. Electron. Telecom. Eng.* **35**(4), 207–217 (1989).
16. J. Khouri and K. A. Stelson, "Efficient algorithm for shortest path in 3-D with polyhedral obstacles," *Trans. ASME – J. Dyn. Syst. Meas. Control* **8**(3), 433–436 (1989).
17. M. Kohler and M. Spreng, "Fast computation of the c-space of convex 2D algebraic objects," *Int. J. Rob. Res.* **14**(6), 590–608 (1995).
18. T. Lozano-Perez', "Spatial planning: A configuration space approach," *IEEE Trans. Comput.* **C-32**(2), 108–120 (1983).
19. T. Lozano-Perez', "A simple motion planning algorithm for general robot manipulators," *IEEE Trans. Rob. Autom.* **RA-3**(3), 207–223 (1987).
20. V. Lumelsky and K. Sun,'A Study of the Obstacle Avoidance Problem Based on the Deformation Retract Technique," *Proceedings of the 29th. IEEE Conference on Decision and Control*, Honolulu, HI (Dec. 1990) pp. 1099–1104.
21. V. Lumelsky and K. Sun, "A unified methodology for motion planning with uncertainty for 2-D and 3-D two-link robot arm manipulators," *Int. J. Rob. Res.* **9**(5), 89–104 (1990).
22. E. Ralli and G. Hirzinger, "Global and resolution complete path planner for up to 6 dof robot manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation*, Minneapolis, MN (Apr. 1996) pp. 3295–3302.

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press

23. R. E. Red and H. V. Truong-Cao, "Configuration maps for robot path planning in two dimensions," *Trans. ASME – J. Dyn. Syst. Meas. Control* **107**, 292–298 (1985).
24. W. E. Red, H. V. Truong-Cao, K. H. Kim, 'Robot path planning in three dimensions using the direct subspace," *Trans. ASME – J. Dyn. Syst. Meas. Control* **109**, 238–244 (1987).
25. D. Roy, "Study on the configuration space based algorithmic path planning of industrial robots in an unstructured congested 3-Dimensional space: An approach using visibility map," *J. Intell. Rob. Syst.* **43**(2–4), 111–145 (2005).
26. E. Sacks and C. Bajaj, "Sliced configuration spaces for curved planar bodies," *Int. J. Rob. Res.* **17**(6), 639–651 (1998).
27. E. Sacks, "Practical sliced configuration spaces for curved planar pairs," *Int. J. Rob. Res.* **18**(1), 59–63 (1999).
28. S. Sachs, S. M. La Valle and S. Rajko, "Visibility-based pursuit – evasion in an unknown planar environment," *Int. J. Rob. Res.* **23**(1), 3–26 (2004).
29. J. T. Schwartz and M. Sharir, "A survey of motion planning and related geometric algorithms," *Artif. Intell.* **37**, 157–169 (1988).
30. J. Slotine, E. Jacques and H. S. Yang, "Improving the efficiency of time-optimal path following algorithm," *IEEE Trans. Rob. Autom.* **RA-5**(1), 118–124 (1989).
31. B. J. H. Verwer, "A Multi-Resolution Workspace, Multi-Resolution Configuration Space Approach to Solve the Path Planning Problem," *Proceedings of the IEEE International Conference on Robotics and Automation,* Cincinnati, OH, USA (1990) pp. 2107–2112.
32. E. Welzl, "Constructing the visibility graph for n-line segments in o (n^2) time," Inf. Process. Lett. **20**, 167–171 (1985).
33. K. D. Wise and A. Bowyer, "A survey of global configuration-space mapping techniques for a single robot in a static environment," *Int. J. Rob. Res.* **19**(8), 762–779 (2000).
34. Y. Yu and K. Gupta, "C-space entropy: A measure for view planning and exploration for general robot – sensor systems in unknown environment," *Int. J. Rob. Res.* **23**(12), 1197–1223 (2004).
35. A. Zelinsky, "Using path transforms to guide the search for findpath in 2D," *Int. J. Rob. Res.* **13**(4), 315–325 (1994).

https://doi.org/10.1017/S0263574709990786 Published online by Cambridge University Press