Robotics and Autonomous Systems 57 (2009) 194–201

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

A variational approach to path planning for hyper-redundant manipulators

Bhaskar Dasgupta*, Akhil Gupta, Ekta Singla

Department of Mechanical Engineering, Indian Institute of Technology, Kanpur–208 016, India

ARTICLE INFO

Article history:
Received 19 March 2007
Received in revised form
30 April 2008
Accepted 6 May 2008
Available online 9 May 2008

Keywords:
Path optimization
Variational calculus
Redundant manipulators
Obstacle avoidance

ABSTRACT

Motion planning for hyper-redundant manipulators in a complicated and cluttered workspace is a challenging problem. Many of the path planning algorithms, based on cell decomposition or potential field, fail due to the high dimensionality and complex nature of the C-space. Probabilistic roadmap methods (PRM) which have been proven to be successful in high dimensional C-spaces suffer from the drawback of generating paths which involve a lot of redundant motion. In this paper, we propose a path optimizing method to improve a given path in terms of path length and the safety against the collisions, using a variational approach. The capability of variational calculus to optimize a path is demonstrated on a variety of examples. The approach succeeds in providing a good quality path even in high dimensional C-spaces.

1. Introduction

A robot manipulator is said to be *redundant* in a general sense, when it possesses more degrees of freedom (DOF's) than required for the task. Redundancy in a manipulator allows more versatile motion and provides greater capability to avoid obstacles and singular configurations [1]. However, inverse kinematics and control of redundant manipulators become increasingly complicated with each added degree of freedom. With a large number of redundant degrees of freedom, as in the so called 'hyper-redundant' manipulators, the motion capabilities in highly cluttered environments can be enhanced to a great extent, but the corresponding path planning problem also become increasingly challenging.

In recent years, efforts have been made to develop efficient path planners for robots with large numbers of DOF's. A number of methods have been proposed for solving the path planning problem [2,3] and most of these use the concept of configuration space (C-space), where the path planning problem is reduced to the determination of a set of connected points in the free C-space from the start configuration to the final configuration [4]. Most of the motion planning algorithms can be broadly classified into three categories: *cell decomposition*, *roadmap* and *potential field* methods. Cell decomposition methods are based on dividing the configuration space into cells such that a sequence of cells connecting the cell containing the initial configuration to that containing the goal configuration can be generated [5,6]. These methods are computationally very expensive and particularly the enormous storage requirement prohibits its use for higher dimensional C-spaces. Roadmap techniques consist of generating a network of one-dimensional paths called a *roadmap* in the configuration space [7]. Such a roadmap consists of free configurations connected to one another. Path planning is thus reduced to searching for a path from the start node to the final node in the network. However, this approach requires a large amount of time, spent in generating the roadmap, which can be a serious drawback in cases where the dimension of the C-space is high. A particular type of roadmap methods, called the *Probabilistic RoadMap Method* (PRM) [8], has proved to be successful in finding feasible paths in a wide range of planning problems involving robots with high DOF's.

A modified PRM method, called *Single-query, Bi-directional, Lazy in collision checking* (SBL), was suggested by Sanchez and Latombe [9]. The algorithm consists of expanding two graphs, one at the start node and the other at the goal node, simultaneously by generating nodes randomly. This is continued until a path is found connecting the two graphs, giving a path from the start node to the final node. The graphs are generated each time a path is required and is particularly suitable for robots working in changing environments. However, due to the stochastic nature of the algorithm, the generated path often involves redundant movements, is far from optimal in any sense and has poor quality. In fact, at times, the resulting path contains so much of wandering around that it is useless for any practical purpose, though theoretically feasible!

Potential field methods use gradient search method to move the robot from one configuration to another. The potential functions are defined in such a manner that the potential is high near the obstacles and low near the goal configuration [10–12]. Thus,

* Corresponding author. Tel.: +91 512 259 7995; fax: +91 512 259 7408.
E-mail address: dasgupta@iitk.ac.in (B. Dasgupta).

0921-8890/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2008.05.001

by proceeding in the direction of decreasing potential, the robot is expected to reach its goal configuration. However, difficulty in formulating the proper potential function to avoid local minima is a problem with this approach. Another method using potential field is the variational approach [13–17]. This strategy is helpful to obtain optimal paths by operating upon the entire path simultaneously. Warren [15] presented the approach which consists of starting with a trial path and improving it in subsequent iterations. The information of the C-space is required *a priori*, which is a major drawback for any path planning method and a barrier to work with higher dimensions. Seshadri and Ghosh [16] proposed the method of local variations to improve an initial guess. A systematic approach of variational dynamic progamming is presented by Barraquand and Ferbach [17]. It starts from a path generated for an obstacle-free space and leads to generation of paths for workcells which are gradually increasing in complexity, towards the path for the actual workcell. The work discussed in these references is surely a step towards global path planning but either has been implemented for workspaces with only circular obstacles or require the C-space description. Also, the robots in use are of very few DOF's.

In this paper, we propose an approach using variational calculus which provides optimal or near-optimal paths in high dimensional C-spaces. This approach has been successfully applied by our group [18] for the problem of singularity avoidance of parallel-actuated manipulators. The issues are different in the present problem, which reflects in several important aspects of the methodology, though the central essence of the variational formulation is common. The method involves the definition of a functional of the path and optimizing it over all possible paths. A feasible initial guess of the solution is helpful for the optimization to perform well. In the present work, the variational approach has been implemented so as to operate with only feasible solutions, in which case such a feasible initial guess is necessary to start the iterations. We have used SBL algorithm to generate a feasible initial guess. With the availability of an initial feasible solution, the performance of the variational approach is greatly enhanced and a penalty formulation against constraint violations is avoided.

In the next section, the pre-processing by the SBL method, a probabilistic roadmap method, is summarized. Section 3 presents the formulation of the problem as a set of differential equations using the variational approach. The solution of the resulting boundary value problem (BVP) is then discussed in Section 4. Section 5 presents a few case studies to plan paths for planar manipulators. Finally, in the last section, contributions of the present work are summarized along with a discussion on possible future extensions.

2. Single-query, bi-directional, lazy in collision detection PRM

In this section, we briefly discuss the Single-query, Bi-directional Lazy in collision detection probabilistic roadmap method (SBL) [9]. It uses probabilistic roadmap technique to generate a path from the start configuration to the goal configuration. A path so generated is feasible but is not optimal in any sense. The planner takes in two parameters: n — the maximum number of milestones that it is allowed to generate and a distance threshold, ρ. Two milestones are considered to be close to each other if the distance between them in configuration space is less than ρ.

The planner builds two trees, with the initial configuration $\mathbf{q}_i$ and the final configuration $\mathbf{q}_f$ as their roots. Expansion of the roadmap is effected by first selecting one of the trees to expand. Next, a milestone m is selected in that tree with probability $P(m) = \frac{1}{d(m)}$ where $d(m)$ is the density of the milestones in the neighbourhood of m. A new collision-free configuration is randomly generated within a distance ρ of the selected node. This is the new milestone. From this new milestone, let m' be the closest milestone added in the other tree. Both trees are connected if there is a collision-free path from m' to m within a distance ρ, else the expansion of the trees goes on. Repetition of this step eventually creates a path joining $\mathbf{q}_i$ to $\mathbf{q}_f$.

A collision-free path between any two nodes is generated by using a local planner. The local planner essentially attempts to join the two points in C-space by a straight line and returns *failure* if it is unable to do so. This straight path is generated by checking whether intermediate points are collision-free or not. In planar manipulators, this is achieved using simple polygon intersection algorithms.[1]

Once a feasible (collision-free) path is constructed in the C-space, we parametrize it with respect to a scalar parameter u over the interval [0,1]. The milestone configurations are uniformly distributed as $\mathbf{q}(u)$ over this normalized interval, with the given initial and final configurations being $\mathbf{q}(0) = \mathbf{q}_i$ and $\mathbf{q}(1) = \mathbf{q}_f$, respectively. Thus, we have a feasible initial (guess) solution for the variational formulation to operate on. Through successive steps of the optimization method, the number of grid points in this discretized path will remain as the number of milestones in the original guess solution, though the optimization procedure will, in general, iteratively shift the configurations from the original 'milestones'.

3. Variational formulation

The primary objective of the path-planning strategy is to find a path completely within the free space connecting the initial configuration ($\mathbf{q}_i$) to the final configuration ($\mathbf{q}_f$), i.e. to find the function

$$\mathbf{q} = \mathbf{q}(u) \quad \text{for } 0 \le u \le 1, \tag{1}$$

where

$$\mathbf{q}(0) = \mathbf{q}_i \quad \text{and} \quad \mathbf{q}(1) = \mathbf{q}_f, \tag{2}$$

such that

$$\mathbf{q}(u) \in \text{Free space} \quad \text{for } 0 \le u \le 1. \tag{3}$$

The variational approach involves formulating the path planning problem as an optimization problem by defining a functional which is to be minimized. In analogy with the notions of classical dynamics, the chosen functional is based on the Lagrangian, defined as

$$L = K(\mathbf{q}, \dot{\mathbf{q}}) - P(\mathbf{q}). \tag{4}$$

Here,

$$K(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{W}\dot{\mathbf{q}} \tag{5}$$

represents a quantity analogous to *kinetic energy*, the incorporation of which tends to keep the path short. A symmetric positive definite matrix $\mathbf{W}$ can be used for scaling the different components of the vector $\mathbf{q}$. In the current implementation, we use identity matrix as $\mathbf{W}$, i.e. all the coordinates here are kept at the same scale.

$P(\mathbf{q})$ is a penalty function, treated as the *potential energy*, which ensures that the path remains within the workspace and does not collide with the obstacles. The potential (penalty) function is defined as

$$P(\mathbf{q}) = \begin{cases} c_1, & \text{if } \mathbf{q} \notin \text{ Free space} \\ c_1 e^{-c_2\left(\frac{d}{d_0}\right)^2}, & \text{if } \mathbf{q} \in \text{ Free space} \end{cases} \tag{6}$$

[1] For spatial manipulators, some intersection checking module, for example RAPID or PQP [19], needs to be used.

where d is the shortest distance of the robot from the obstacles or that among links of the robot at configuration $\mathbf{q}$ and d_0 is a non-dimensionalizing constant.

The constant c_1 is chosen to be a very large number, so that the potential due to obstacle collision is very high. (For most of the cases discussed below, we used $c_1 = 10^{30}$.) This is necessary to maintain the path in free space as no collision is to be allowed during the optimization. As will be apparent shortly, the potential function enters the actual computation only through its gradient. With the above definition of $P(\mathbf{q})$, at a configuration in the interior of a C-obstacle, the gradient turns out to be zero, which would fail to influence the iterative process. This makes an infeasible configuration unacceptable in any iteration. A large value of c_1 penalizes an approach to collision so strongly that feasibility is never compromised during the iterations.

The potential in the neighbourhood decreases exponentially till it reduces to zero at points fairly within the free space. The rate of this decrement is controlled by the constant $c_2 > 0$, together with d_0. In our examples, we use $c_2 = 1$ and $d_0 = 0.1$. The role of these two parameters could be accomplished by a single variable. However, we maintain the two parameters separately to retain greater flexibility. Parameter c_2 is related closely with the *method*, while d_0 is closely associated with the user's data, particularly their *units*!

By minimizing the integral $\int_0^1 L\mathrm{d}t$ we get the Euler–Lagrange equations in the form

$$\mathbf{W}\ddot{\mathbf{q}} = -\frac{\partial P}{\partial \mathbf{q}}. \tag{7}$$

This equation, along with the boundary conditions from Eq. (2), forms a two-point boundary value problem. A brief discussion on the solution process of the resulting boundary value problem (BVP) is presented in the next section.

It can be noted that the resulting ODE (ordinary differential equations) system is autonomous, i.e. they do not have explicit dependence on the path parameter u, which is the independent variable here. As such, a rescaling of the parameter domain, i.e. a reparametrization, would have no impact on the result, as long as a consistent parametrization is used for representation of the path and handling of the finite difference equations discussed below. The parameter interval [0,1] used in the above formulation is only for simplicity of the formalism. In practice, any suitable interval can be used.

4. Solving boundary value problems: Relaxation method

There are two widely used classes of numerical methods for solving two-point boundary value problems, namely *shooting* and *relaxation methods* [20,21]. In the former method, values for all the dependent variables are chosen at one boundary. These values should be consistent with the prescribed boundary conditions. The system of ODE's is then solved by using initial value methods, arriving at the other boundary. In general, there will be discrepancies with the boundary conditions at the other end. The free parameters at the starting point are adjusted iteratively so as to eliminate these discrepancies. Relaxation methods use a different approach. Differential equations are replaced by finite-difference equations (FDE's) on a mesh of points that covers the domain of integration. A trial solution consists of values for the dependent variables at each mesh-point, possibly *not* satisfying the finite-difference equations, nor necessarily even satisfying the boundary conditions. The iteration, now called *relaxation*, consists of adjusting all the values on the mesh so as to bring them in closer agreement with the finite difference equations and boundary conditions.

In general, shooting method is computationally more intensive than the relaxation method and does not guarantee convergence to a solution. On the other hand, for the relaxation method to work, an initial solution for the entire domain must be provided. As described earlier, in this work, SBL is used as a pre-processor which provides the initial feasible solution. Using this path as an initial guess, the relaxation method is used to find the final quasi-optimal solution. We call the final solution *quasi-optimal*, because it is optimal in the sense of the 'subjective' definition of optimality through the development of the Lagrangian.

In order to solve the ODE system (Eq. (7)) numerically, we first convert them into a system of first order differential equations. Let

$$\mathbf{Y} = \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}u} \quad \text{and} \quad \mathbf{Z} = \begin{bmatrix} \mathbf{q} \\ \mathbf{Y} \end{bmatrix}. \tag{8}$$

We obtain the system of $2N$ first order ordinary differential equations as

$$\frac{\mathrm{d}\mathbf{Z}}{\mathrm{d}u} = \begin{pmatrix} \mathbf{Y} \\ -\mathbf{W}^{-1}\dfrac{\partial P}{\partial \mathbf{q}} \end{pmatrix}. \tag{9}$$

In relaxation method, we first replace ODE's by approximate *finite difference equations*. For example, a general first-order differential equation

$$\frac{\mathrm{d}z}{\mathrm{d}u} = g(u, z) \tag{10}$$

can be replaced by an algebraic equation relating function values at points k, $k-1$ as

$$z_k - z_{k-1} - (u_k - u_{k-1})\, g\left(\frac{u_k + u_{k-1}}{2}, \frac{z_k + z_{k-1}}{2}\right) = 0. \tag{11}$$

When the problem consists of N coupled first-order ODE's represented by FDE's on a mesh of M points, a solution consists of the values of N dependent functions at each of the M mesh points, i.e. $N \times M$ variables in all. Let $\mathbf{z}_k$ denote the vector of dependent variables $z_1, z_2, \ldots, z_N$ at point u_k. At an arbitrary point u_k, the set of N first order equations are of the form

$$0 = \mathbf{E}_k \equiv \mathbf{z}_k - \mathbf{z}_{k-1} - (u_k - u_{k-1})\, \mathbf{g}_k\left(\frac{u_k + u_{k-1}}{2}, \frac{\mathbf{z}_k + \mathbf{z}_{k-1}}{2}\right), \quad k = 2, 3, \ldots, M. \tag{12}$$

At the first boundary, the boundary conditions give

$$0 = \mathbf{E}_1 \equiv \mathbf{B}(u_1, \mathbf{z}_1); \tag{13}$$

while at the second boundary,

$$0 = \mathbf{E}_M \equiv \mathbf{C}(u_M, \mathbf{z}_M). \tag{14}$$

Thus, the $2N$ equations at points k take the forms

$$0 = E_{j,k} \equiv z_{j,k} - z_{j,k-1} - (u_k - u_{k-1})\frac{(z_{N+j,k} + z_{N+j,k-1})}{2}, \quad j = 1, 2, \ldots, N \tag{15}$$

and

$$0 = E_{j+N,k} \equiv z_{j+N,k} - z_{j+N,k-1} + (u_k - u_{k-1})\frac{\partial P}{\partial z_j}, \quad j = 1, 2, \ldots, N; \tag{16}$$

in which $\mathbf{W} = \mathbf{I}$ has been used. At the first boundary,

$$0 = \mathbf{E}_1 \equiv \mathbf{q}(0), \tag{17}$$

with $\mathbf{q}(0)$ as the starting configuration of the robot. Similarly, at the second boundary,

$$0 = \mathbf{E}_M \equiv \mathbf{q}(1), \tag{18}$$

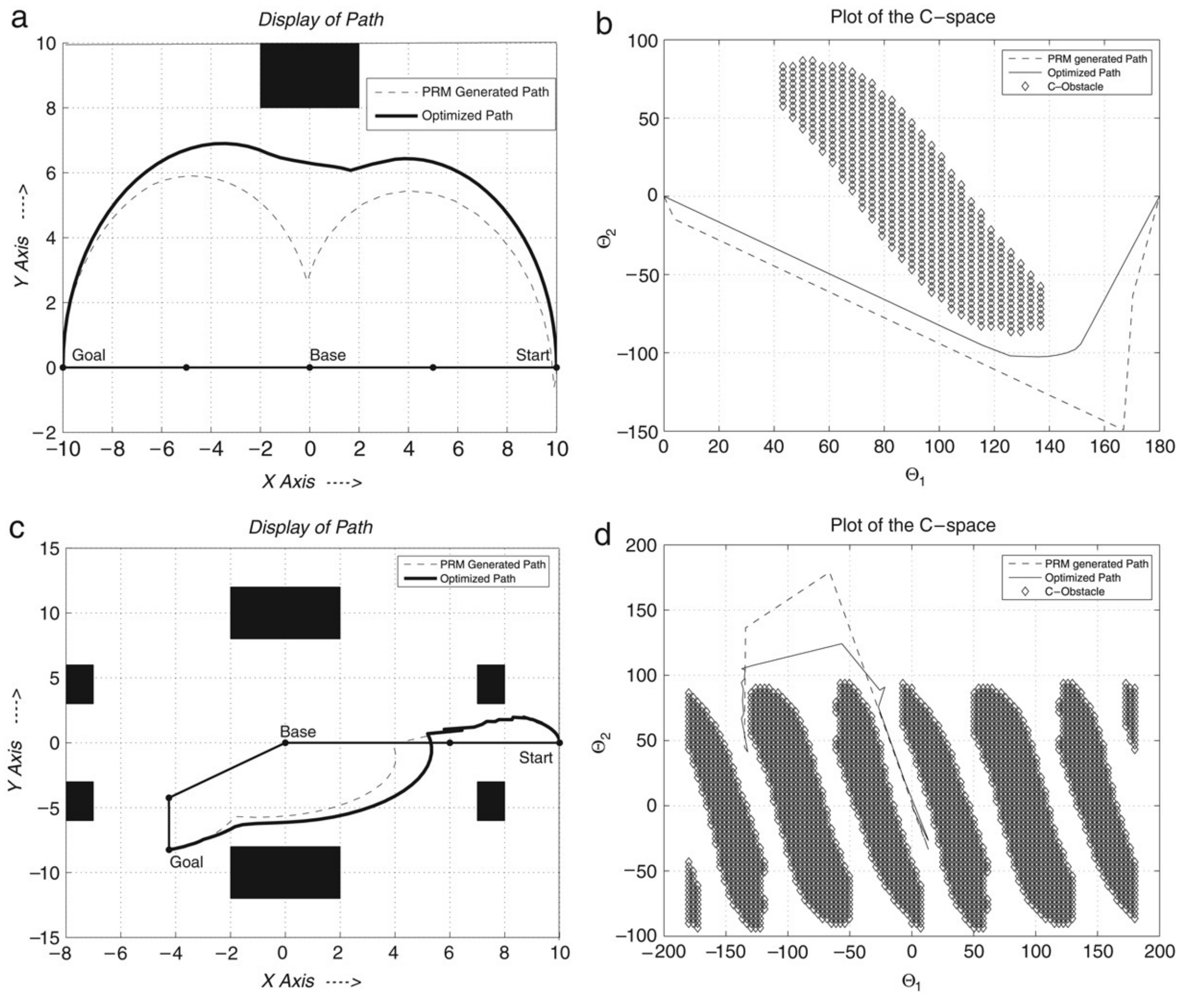

Fig. 1. (a) and (b) show the results for a 2-DOF manipulator with 1 obstacle and (c) and (d) show the results obtained for another 2-DOF manipulator with 6 obstacles.

with $\mathbf{q}(1)$ as the final configuration of the robot. These provide additional equations to complete the system.

The methodology, in general, allows the algorithm to apply corrections to all the variable values, even at the boundary points. But, this is not desirable in our problem where the solution at the boundary points represent the start and the goal configurations. To prevent this, we have allowed no correction to the values of the boundary point solutions and regarded them as constants.

Due to the complex nature of the potential function in crowded workspaces, a very accurate solution is often difficult to obtain. However, this is not a problem as we are essentially searching for feasible solutions with reasonably good characteristics, the optimization formulation being only a means to that end.[2] In addition to the convergence threshold, we also have a limit on the maximum number of iterations allowed.

5. Simulation results

The process discussed in the previous section is implemented in a C program and tested for planar manipulators of different DOF's. In robot specifications, base point, number of links, link lengths and allowed joint variable limits are provided. The workspace is specified in the form of a number of polygonal obstacles and each obstacle is described with an ordered set of vertices. The main performance criterion is the length of the path in configuration space obtained by the variational approach as compared to that generated by SBL. Another important metric is the minimum distance of the robot from the obstacles along the path. The potential field prevents the manipulator from coming very close to the obstacle. That is, iterations of the relaxation method maintain a margin from collision as long as the initial path is collision-free. Table 1 summarizes the results presented in the following. The units of length in this work are taken as centimeters. The width of each link in all the cases is taken as 2 units.

Although a manipulator with two degrees of freedom seems trivial, we first use it in our simulations for easier visualization of the results in C-space. Fig. 1 shows the results for 2-DOF manipulators for two different workspaces. The link lengths of the first manipulator are 5 units each and those of second are 6 and 4 units. Fig. 1(a) and (c) present the PRM and optimized path for the two cases in Cartesian space. These figures also show the initial and the final configurations. The optimization of the path is clearly evident in Fig. 1(b) and (d), showing the results in C-space for the two cases. It is clear from these figures how the length of the path in joint space, measured in degrees, has been reduced, in both cases. Since it is difficult to present the C-space for higher dimensions, for the rest of the examples discussed in this paper, this reduction of length from initial guess to final path is shown in third and fifth column of Table 1. The manipulators used in these case studies consist of revolute joints only and hence the

[2] The reader would easily appreciate that insisting on a formally exact optimal solution is not very productive, since the definition of optimality itself is qualitative here.

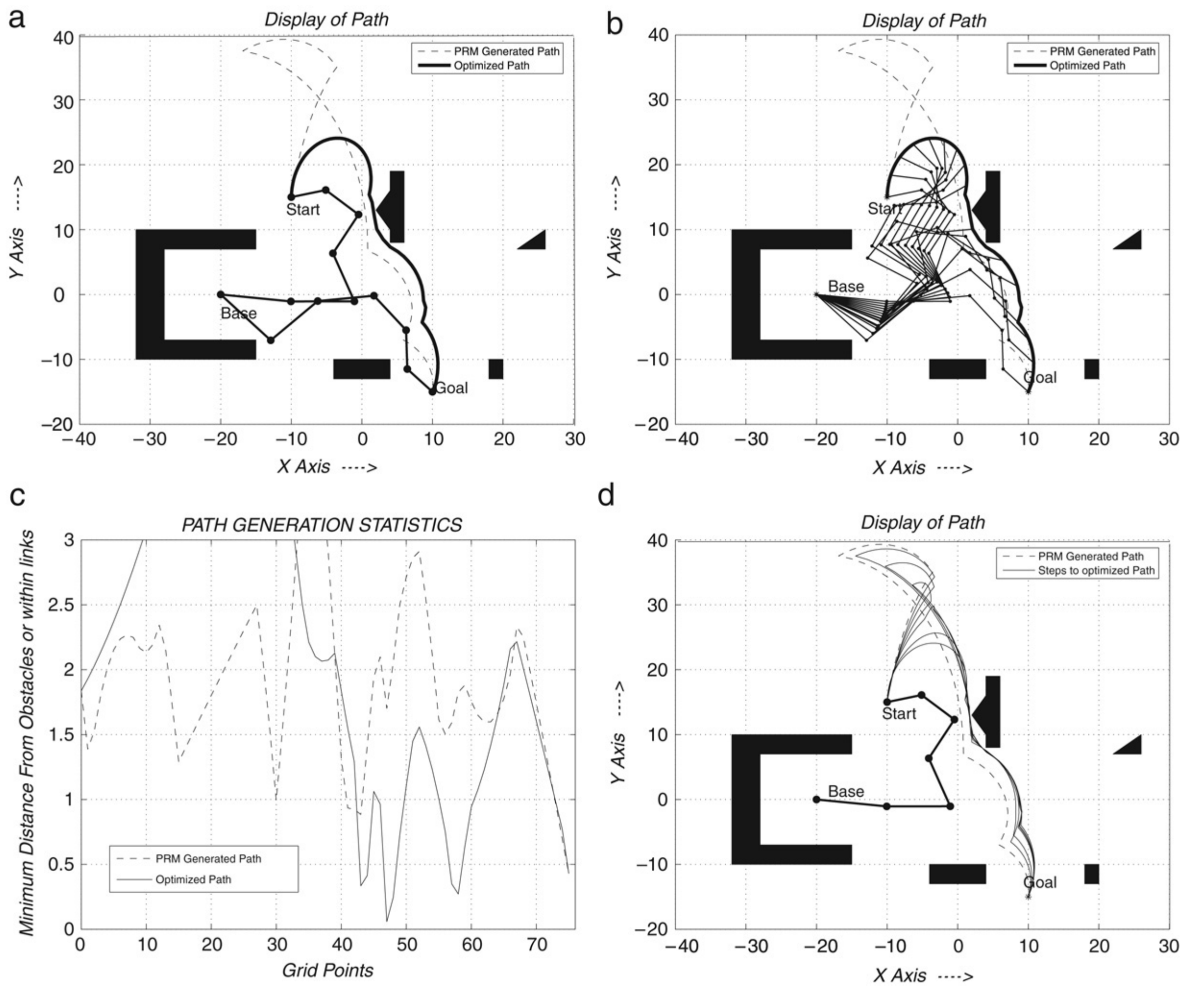

Fig. 2. 6-link manipulator with 5 obstacles. (a) The paths. (b) Manipulator tracking the optimal path. (c) Minimum distance. (d) Stepwise change from initial path to final path.

Table 1
Result statistics for all presented cases

Degrees of freedom	Number of obstacles	SBL		Variational		Number of iterations
		Path length	Least distance	Path length	Least distance	
2	1	376.67	1.76	289.29	0.82	29
	6	321.46	0.287	326.55	0.436	32
6	5	686.76	0.43	367.12	0.06	11
8	3	476	0.34	238.34	0.003	21
	5	2434	0.06	829.21	0.16	31
10	4	561.26	0.35	268.43	0.4	29
12	4	813.87	0.22	604.68	0.32	25

path length in the joint space is easily reckoned in consistent units, namely degrees. However, if the manipulator has any prismatic joint(s), then the length of the path (in joint space) will need to be suitably normalized.

Fig. 2(a) shows paths obtained for a 6-dof manipulator with 6 obstacles. The link lengths of the manipulator are [10, 9, 8, 7, 6, 5]. The variational approach has succeeded in optimizing the path. Fig. 2(b) shows some intermediate steps of the robot while tracking the optimal path. It is clear from the figure that the robot avoids collision with the obstacles. But, to be sure about no collision even within links, Fig. 2(c) has been included. This plot shows the minimum distance of any link of the robot from either any of the obstacles or from the other links, along the path. All distances are found to be positive, thereby proving the collision avoidance of the path. Stepwise change in the path (of the end-effector) from the given initial one to the optimal one through the relaxation iterations is presented in Fig. 2(d), showing the manner of progress of the iterations. These are a few intermediate steps taken out of all the iterations of the optimizing method.

Similar plots for an 8-link manipulator are shown in Fig. 3 for a workspace with 3 obstacles. The link lengths of the manipulator are [8, 8, 7, 7, 6, 6, 6, 6]. Fig. 3(a) shows the paths that are traced by the end-effector point and (b) shows some steps of path-tracking by the entire manipulator. This is the only case presented in the paper, in which a compromised value of $c_1 = 10^{15}$ is used instead of the default value of 10^{30}. This explains how the algorithm allows such a close approach to an obstacle and that too over a significant interval of the path, as evident from Table 1 and the minimum distance plot of the optimized path in Fig. 3(c) also. Obviously, this can be improved by executing the variational approach with

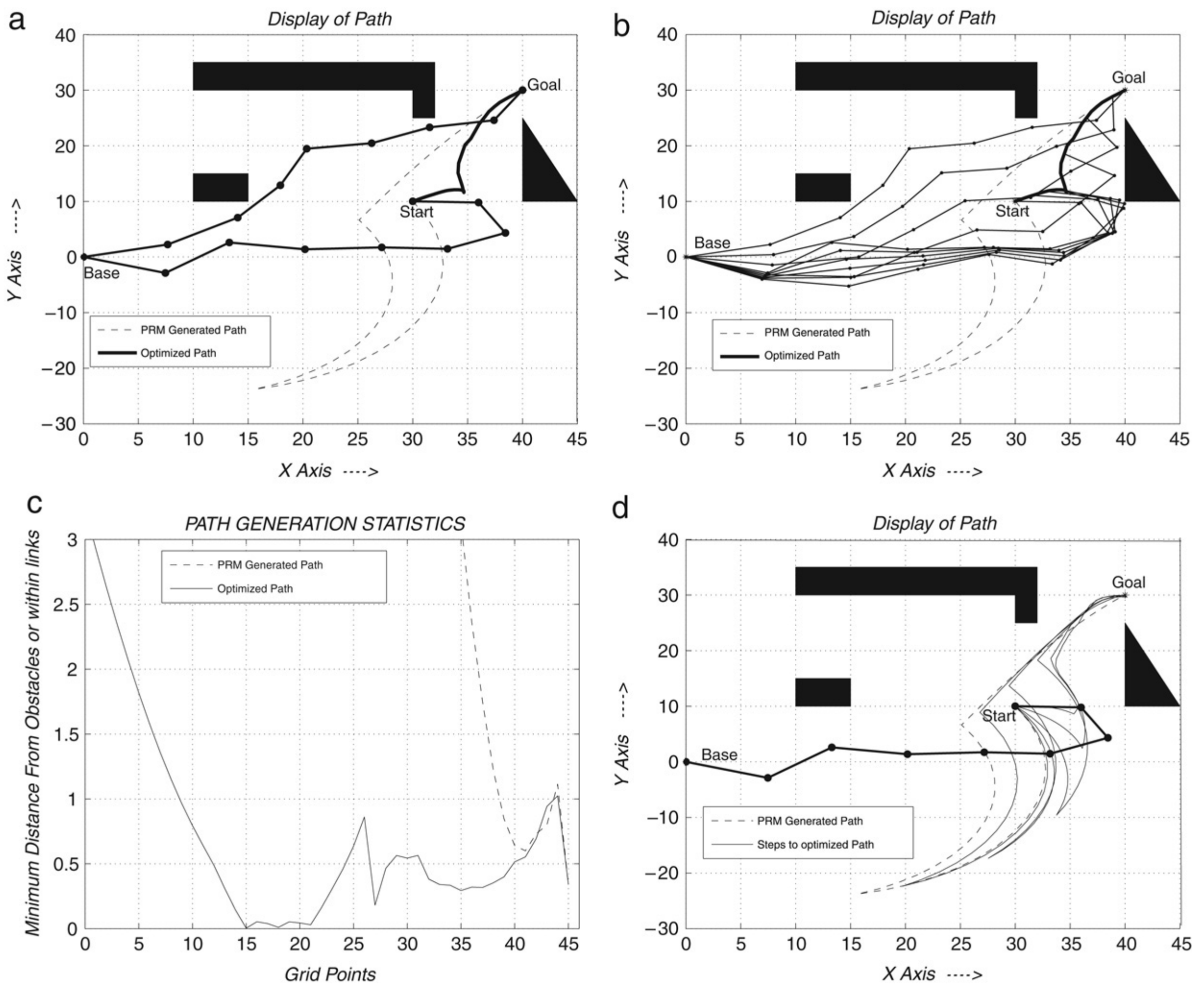

Fig. 3. 8-link manipulator with 3 obstacles. (a) The paths. (b) Manipulator tracking the optimal path. (c) Minimum distance. (d) Stepwise change from initial path to final path.

a higher value of c_1. Our verification shows that using $c_1 = 10^{30}$, the distance of closest approach increases from 0.003 to 0.19 unit. It can be mentioned in this context that the notion of *optimality* in this work pertains to an appropriate trade-off between two requirements of short path length (in C-space) and maintenance of margin from collision. The extent of their relative importance is controlled by the parameter c_1. Other than taking extremely high value for c_1, another way of maintaining a minimum margin is to take some tolerance width for the links in the beginning itself so as to practically avoid the obstacles by more significant distance. Stepwise change in the paths, as shown in Fig. 2(d), furnishes additional information on the path segment of closest approach and corresponding neighbouring objects.

Fig. 4 shows results for another 8-link manipulator with 5 obstacles. This result is included to show the redundancy in the SBL path. The substantial improvement given by the optimization approach proves the power of the proposed method.

Two more cases with 10-DOF and 12-DOF manipulators are shown in Figs. 5 and 6. The workspace used is same for the two cases, but the start positions and the end positions are different. Besides presenting further evidence of effectiveness of the variational approach for higher DOF's, these cases also demonstrate an important issue. The variational formulation and the subsequent relaxation iterations typically preserve the topology of the initial guess path in the context of the obstacles. This inability to jump to a topologically different path is, indeed, a minor limitation of the method. However, several runs of SBL method, which is not expensive, can provide several paths. These paths can be optimized through variational approach to obtain a number of paths with different topologies. Considerable diversity of path topology can be achieved in this manner, though the process is not exhaustive.

6. Conclusion

A variational calculus based path planner for hyper-redundant manipulators is presented in this paper. The approach combines the roadmap based SBL algorithm and potential based variational technique to generate feasible and good quality paths. SBL provides the variational algorithm with an initial feasible solution. Optimization of this path, using the relaxation method, generates a near-optimal path. Results for several planar manipulators have been presented which demonstrate the efficiency of the algorithm. The approach succeeds in reducing the length of the path and generating a smoother path for the manipulator. The optimization is more strongly visible in cases with higher number of degrees of freedom. This is partly due to the increasing dimension of the C-space, for which the randomness in PRM generated path increases.

The approach can be extended to spatial manipulators, for which the notable additional work is not on the method as such, but mostly for the collision detection and distance evaluation between 3-dimensional objects. Incorporation of distance functions and proximity analyses among solid objects, as in [22,19], can be useful for the purpose. Another important extension of the

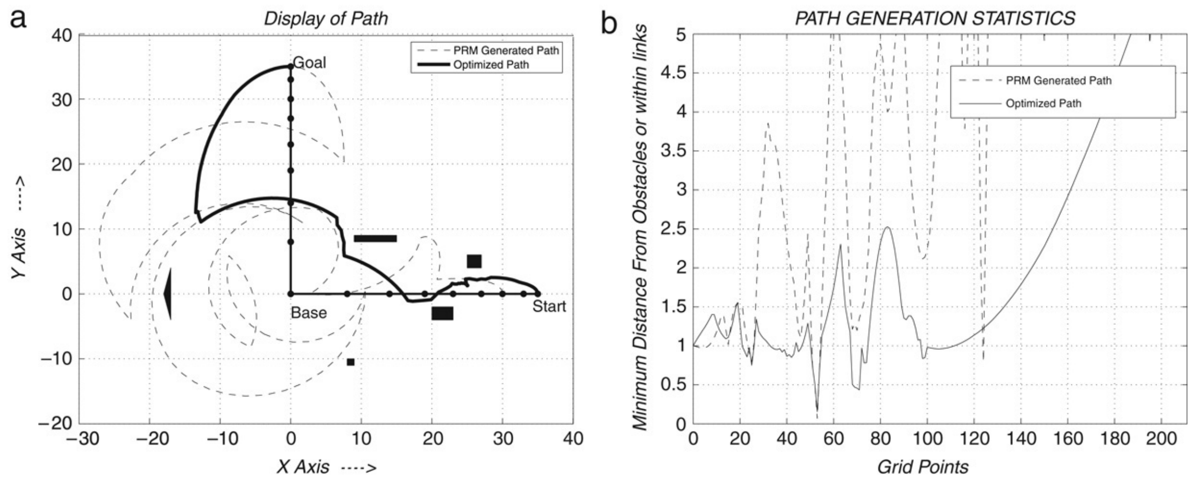

Fig. 4. 8-link manipulator with 5 obstacles. (a) The paths. (b) Minimum distance.

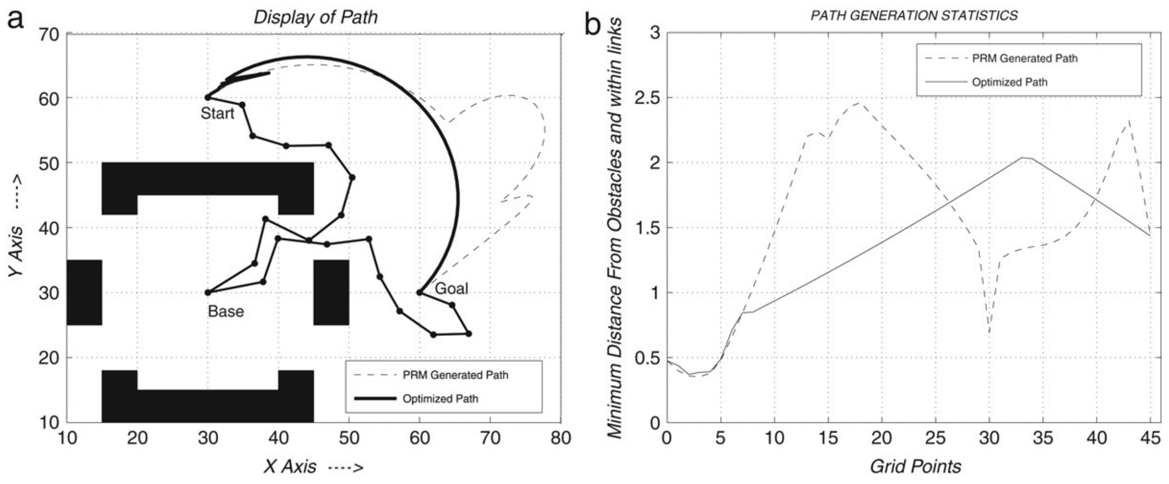

Fig. 5. 10-link manipulator with 4 obstacles. (a) The paths. (b) Minimum distance.

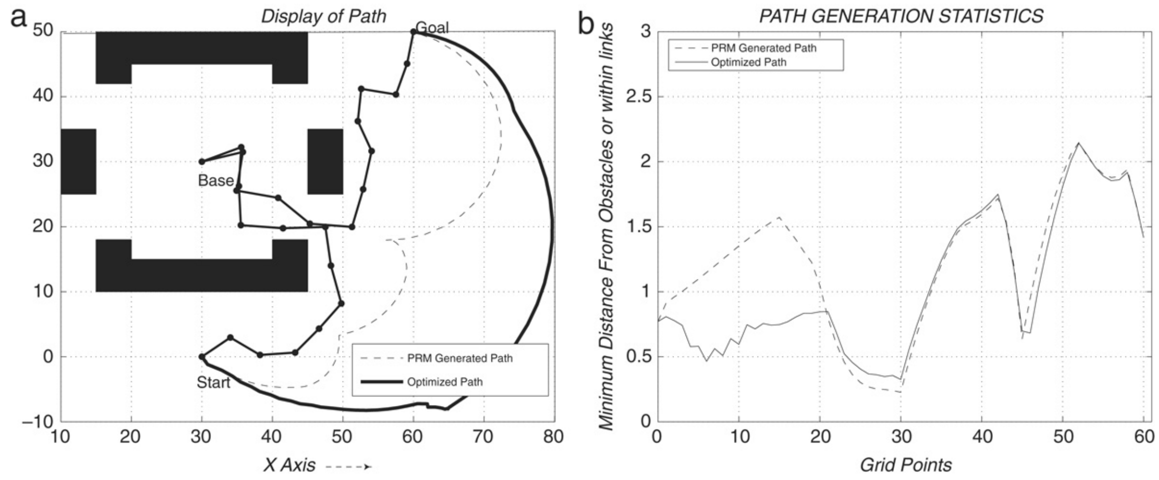

Fig. 6. 12-link manipulator with 4 obstacles. (a) The paths (b) Minimum distance.

approach is to enhance its capability to handle infeasible configurations as well through a smooth penalty. By assigning a variable penalty based on the depth of collision, one can completely avoid the necessity of the pre-processing step of developing a feasible path. With a first order continuous penalty function, quantitatively reflecting the *extent* of infeasibility, the robust method of relaxation is expected to converge, starting even from a grossly inaccurate and infeasible initial solution. As found in [18] in a different context, a direct linear interpolation of the initial and final configurations can be then used as the initial guess.

Acknowledgement

Authors gratefully acknowledge the Govt. of India Grant No. 2000/36/2-BRNS for partial financial support for this work.

References

[1] T. Yoshikawa, Manipulability of robotics mechanisms, Journal of Robotics Research 4 (2) (1985) 3–9.
[2] J.C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, 1990.
[3] J. Schwartz, Planning, geometry, and complexity of robot motion, in: Intellect Books, 1989.
[4] P. Lozano, Spatial planning: A configuration space approach, IEEE Transactions on Computers 32 (2) (1983) 108–120.
[5] F. Avnaim, J.D. Boissonnat, B. Faverjon, A practical exact motion planning algorithm for polygonal object amidst polygonal obstacles, in: Proceedings of the Workshop on Geometry and Robotics, 1988, pp. 67–86.
[6] F. Lingelbach, Path planning using probabilistic cell decomposition, in: Proceedings International Conference on Robotics and Automation, 2004. p. 1.
[7] C. O'Dunlaing, M. Sharir, C.K. Yap, Retraction: A new approach to motion planning, in: Proc. 15th Annu. ACM Sympos. Theory Comput. 1983, pp. 207–220.
[8] L. Kavraki, P. Svestka, J.C. Latombe, M.H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (4) (1996) 566–580.
[9] G. Sanchez, J.C. Latombe, A single-query bi-directional probabilistic roadmap planner with lazy collision checking, in: Int. Symp. Robotics Research, 2001.
[10] Y.K. Hwang, N. Ahuja, A potential field approach to path planning, IEEE Transactions on Robotics and Automation 8 (1) (1992) 23–32.
[11] J.H. Chuang, Potential-based modeling of three-dimensional workspace for obstacle avoidance, IEEE Transactions on Robotics and Automation 14 (5) (1998) 778–785.
[12] E. Conkur, Path planning using potential fields for highly redundant manipulators, Robotics and Autonomous Systems 52 (2–3) (2005) 209–228.
[13] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, The International Journal of Robotics Research 5 (1) (1986) 90.
[14] S.H. Suh, A variational dynamic programming approach to robot path planning with a distance-safety criterion, IEEE Journal of Robotics and Automation 4 (3) (1987) 334–349.
[15] C. Warren, Global path planning using artificial potential fields, in: Proceedings. IEEE International Conference on Robotics and Automation, 1989, pp. 316–321.
[16] C. Seshadri, A. Ghosh, Optimum path planning for robot manipulators amid static anddynamic obstacles, IEEE Transactions on Systems, Man and Cybernetics 23 (2) (1993) 576–584.
[17] J. Barraquand, P. Ferbach, Path planning through variational dynamic programming, in: Proceedings. IEEE International Conference on Robotics and Automation, 1994, pp. 1839–1846.
[18] S. Sen, B. Dasgupta, A.K. Mallik, Variational approach for singularity-free path-planning of parallel manipulators, Mechanism and Machine Theory 38 (11) (2003) 1165–1183.
[19] S. Gottschalk, M.C. Lin, D. Manocha, OBBTree: A hierarchical structure for rapid interference detection, in: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, 1996, pp. 171–180.
[20] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C^{++}, 2nd edition, Cambridge University Press, 2003.
[21] B. Dasgupta, Applied Mathematical Methods, 1st edition, Pearson Education, 2006.
[22] E. Gilbert, D. Johnson, Distance functions and their application to robot path planning in the presence of obstacles, IEEE Journal of Robotics and Automation 1 (1) (1985) 21–30.

Bhaskar Dasgupta received his bachelor's degree in Mech Engg from Patna University, Patna, India (1991), master's degree in Mech Engg from Indian Institute of Science, Bangalore, India (1993) and Ph.D. from Indian Institute of Science, Bangalore, India (1997). Since 1997, he is teaching at Indian Institute of Technology, Kanpur, India in the Department of Mechanical Engineering, where currently he is holding the position of Associate Professor. His teaching and research interests include robotics, CAD, optimization, applied mathematics and computational biology.

Akhil Gupta is currently employed as a Software Engineer in Google Inc. He completed his Masters in Computer Science in University of Mayrland, College Park, where he conducted research in processing streaming XML data. Prior to this, he obtained his Bachelor of Technology in Computer Science from IIT Kanpur. His area of interests include very large distributed systems, databases and machine learning.

Ekta Singla is currently pursuing her Ph.D. degree in Robotics from Indian Institute of Technology Kanpur, India. She received her Masters of Engineering degree in January, 2002, from Thapar University, Patiala, India. She worked there as lecturer from January 2002 to July 2004. Her areas of interest are robotic design, robot path planning, optimization techniques and computer aided design.