Specialized PRM Trajectory Planning For Hyper-Redundant Robot Manipulators

MAHDI F. GHAJARI and RENE V. MAYORGA
Department of Industrial Systems Engineering
University of Regina
3737 Wascana Parkway, Regina, Saskatchewan
CANADA
mfallahinejad@gmail.com and Rene.Mayorga@uregina.ca

Abstract: - This Paper presents a general comprehensive computationally tractable collision free path planner for hyper-redundant manipulators multiple-query. The path planner is based on a sampling approach for obstacle avoidance and 2D trajectory planning for N-DOF robotic arms. The quality of created roadmap depends on the initializing parameters such as density as well as resolutions of occupancy grid and tip distribution grid. The motion planner has been developed for all types of manipulators, different joint types and handling various cost functions. Various scenarios have been successfully simulated for manipulators with different degrees of freedom and different types of obstacles with any given start and goal configurations. The simulation results indicate that the collision free motions in highly-constrained environments can be computed in less than a minute.

Key-Words: - Optimal Path Planning, Hyper-redundant Manipulators, Obstacle Avoidance, Trajectory Planning

1 Introduction

Motion planning is the process of breaking down an arbitrary robot motion task into computed discrete movements while satisfying motion constraints and optimizing some aspects of the movement. In another word, motion planning in robotics can be defined as the process of generating a path between two pre-defined configurations while avoiding collisions with a set of stationary obstacles [1].

Kinematic redundancy defines as of possession more degrees of freedom than those required to execute a given task. This condition yields to a greater level of dexterity and flexibility for avoiding singularities, joint limits, workspace obstacles while minimizing joint torque and energy consumption level which can be summarized in optimization of performance indexes [2-3].

Generating a collision-free trajectory for an object manipulation from a *start* configuration to a *goal* configuration has been proved to be a hard problem since the complexity of motion planning increases significantly as the number of DOFs grows [4]. Sampling-based motion planning algorithms have been proved to be a successful method in high dimensional C-space and with minimum probability of failure [8].

Although there are various types of trajectory designing algorithms, there is no conclusive method, which could be able to perform in all regions with different types of constraints [5].

Bohlin et al [6] introduced an algorithm based on PRMs called Lazy PRM. This approach is to minimize the running time of the planner by minimizing the number of collision checks performed during planning. The planner constructs a roadmap assuming that all nodes and edges in the roadmap are avoiding obstacles. Song et al [7] presented a customized version of PRMs method that postpone some of the validation checks such as collision check to the query stage which performs efficiently for many problems.

Based on the literature review and to address the drawbacks of the studied methods, a set of research objectives are defined as follows:

- Developing a comprehensive trajectory planner for different scenarios using non-redundant, redundant and hyper-redundant manipulators.
- A novel method for mapping all types of obstacles and workspaces.
- Handling various cost functions based on the scenarios.
- User-control accuracy and resolution based on the scenarios.
- Fast and efficient motion planner.
- High reliability in obstacle avoidance applications.

2 Methodology

The proposed approach for developing a collision-free trajectory planning for redundant and hyper-redundant manipulators in this Paper, is based on sampling-based motion planning and can be divided into two phases: pre-processing phase and query phase.

2.1. The Preprocessing Phase (off-line)

During the preprocessing phase as an off-line stage, a data structure (or roadmap) is constructed in a probabilistic way for a given workspace. The graph contains nodes which are chosen over C_{free} and the edges that correspond to the feasible trajectories. A proper sampling method is needed to generate collision-free configurations in C_{free}. A uniform generation of nodes continues until the desired density is achieved. Each generated random configuration needs to be checked by the collision detector method, whether the random generated configuration is in collision with obstacles. Once random nodes are created, they are connected to their neighboring nodes. The local planner computes a collision free trajectory between a node and its neighbors. Then, randomly created configurations and their connections are added to the graph. In this stage, a roadmap generated in a way that can quickly handle future queries [9].

The Initial empty graph includes only occupied regions. Then, repeatedly, a random free configuration is created and added to the number of nodes. As the planner generates nodes, the local planner connects the generated node to the neighboring nodes with a straight line representing the feasible trajectory between these two nodes. For mapping the obstacle the occupancy grid has to be created. The workflow of pre-processing phase is depicted in Figure 1.

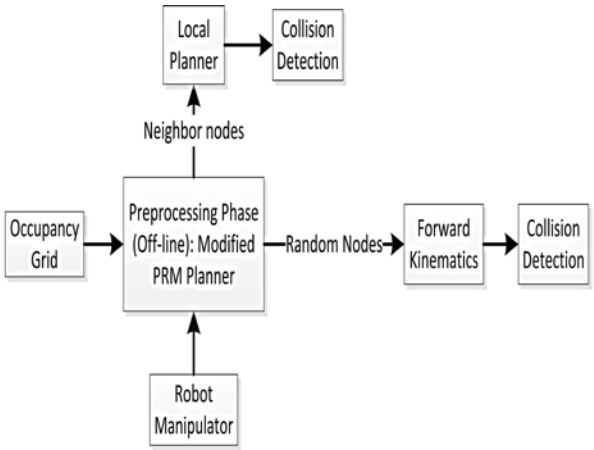


Figure 1. Workflow of the pre-processing phase

1) Robot Manipulator

The first step in the off-line phase is defining the robot manipulator with all possible details. These details includes the number of degrees of freedom (DOF), Denavit-Hartenberg D-H parameters, joint type, joint limit, and link cost values (pre-defined weight for each link).

2) Occupancy Grid Mapping

Trajectory planning requires both manipulator and workspace information. The manipulator information can be safely defined as the number of degrees of freedom, number of links and their dimension. The workspace information consists of obstacles location and their dimension. To create occupancy grid, a numerical value is assigned to each cell that indicates the probability of the cell existence and also the difficulty of its reachability [8]. In another word, the basic idea of the occupancy grid mapping is to represent a map of the workspace with a binary value to represent whether the cell is occupied by the obstacle or it is vacant.

Discretization resolution of manipulator workspace defines the number of cells per axis and the quality of approximation. The occupied cells are listed in C_{obs} and marked with color blue on the roadmap (Figure 7).

The Point in Polygon (PIP) method in general uses repeated geometric queries. Given multiple polygons and a sequence of query points, PIP finds the status for each query point. Each cell calls for PIP function if any vertex of the cell is in the defined polygon. If the answer value is “1” then that cell is considered as an occupied cell and marked with color blue. An extension has been added to this method that adds all of surrounded cells of each occupied cell to the blocked area.

3) Forward Kinematics Calculation

In this Paper, the D-H convention for forward kinematics (FK) calculation is used to compute the position of the links and the end-effector [10]. It is prohibitively difficult to explicitly calculate the shape of C_{free}; however, detecting whether a given configuration is in C_{free} is an efficient solution to this problem. First, the manipulator configuration is calculated and then the collision detector recognizes if the manipulator intersects any of the obstacles.

Equation 1 expresses the location and orientation of the end-effector frame with respect to the base frame as:

$$T_{0n} = A_{01}(q_1)A_{12}(q_2)A_{23}(q_3)\ldots A_{n-1,n}(q\_n) \quad (1)$$

4) Random Sampling

According to Lavalle [8], C_{space}, the configuration space, is "uncountably" infinite while a sampling-based motion planning method can consider at most a countable number of random samples. This method can run forever then it may be "countably" infinite, but practically the expectation is to terminate the algorithm after considering a finite number of random samples. Therefore, the planner generates a dense random configurations for any bounded C_{space}. The maximum number of nodes for each scenario and needed roadmap can be equated to the mass, therefore, the total number can be computed as follows:

Number of Total Nodes = Density × Volume of C_{free} (2)

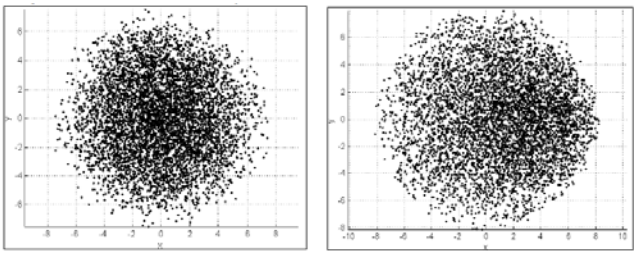
One of the main issues that arises upon random number generation is the probability of distribution. To uniformly distribute all over the workspace, a simple random number generator will not satisfy the problem because it will not uniformly distribute as degrees of freedom increase. In Figure 2.a, a 7-dof manipulator with the maximum length of 8.25 units was chosen to illustrate the end-effector position distribution in black dots. Not only the normal random generator has densely generated samples around the center but also the manipulator end-effector has barely reached the workspace boundaries. Hereby, the shape of the distribution needs to be controlled. Using Beta distribution as a family of continuous probability distribution that is parameterized by two positive shape parameters, denoted by α and β, the result is much closer to a uniform randomized node generation. Figure 2.b indicates the effect of Beta distribution that allows a uniform distribution all over the workspace and reachability of the boundaries.

5) Tip Distribution Grid

According to Figure 3, although the Beta distribution with tuned parameters yields to a more uniform distribution of manipulator's end-effector, there is still a dense region of generated nodes around center towards the right side of the map.

Therefore, the tip distribution grid (TDG) solution is applied to satisfy the distribution problem. The TDG controls the random generation of configurations by controlling the position of manipulator end-effector to uniformly distribute it over the entire workspace. Each TDG cell has a desired and predefined limit of number of nodes. Hence, if the limit is exceeded by the generated random configurations, the generated node will be discarded and the iteration will continue to spread nodes until all TDG cells reach the maximum number of nodes. The maximum number of nodes for each TDG cell is considered as a "mass", thus it can be assumed that the density multiplied volume can be calculated as follows:

Maximum Nodes per Cell = Density × TDG Cell Volume (3)



(a) (b)

Figure 2.
(a) End-effector position distribution of 6000 generated random configurations for 7-dof manipulator (maximum length 8.25 units) using simple random generator.
(b) End-effector position distribution of 6000 generated random configurations for 7-dof manipulator (maximum length 8.25 units) using Beta distribution, $\alpha, \beta=5$

6) Collision Detection

Once a random configuration is generated and the location of the created sample is determined, the collision status of the configuration should be investigated. Hence, collision checking is a critical component of sampling-based planning [8]. The collision detector checks if the generated configuration intersects any part of C_{obs}. If it is collision-free, it will be added to the total generated collision-free nodes; otherwise, it will be discarded. The first step is the calculation of the end-effector's position. Then, this position is sent as a point to the occupancy grid. If the point lies in C_{obs} that point will automatically be discarded. Once the collision detection process is done and the generated configuration's end-effector position is in C_{free}; then we need to check if any part of the robot collides with an obstacle. The assumption during this process is that the robot cannot intersect itself or the links have an appropriate offset to avoid this collision.

7) Finding Neighboring Samples

Once all random nodes are created and the collision detection is performed for each one of

them, they can be considered as the valid nodes and their configuration is placed in the C_{free}. Next crucial step is to find the candidate neighbors around each node. Each connection (edge) in the graph indicates a pair of neighbor nodes (Figure 3). As the density of the roadmap increases, the TDG cells decreases in size and the number of neighboring nodes around each node increases as well. One of the critical factors in sample-based path planning algorithms is the process of pairing close nodes. A criterion (i.e. distance) is required to determine a neighborhood region for each node and then for their connection. The strategy that has been used to find the neighbor nodes in the proposed methodology is the Tip Distribution Grid (TDG). Not only the applied TDG technique solves the problem of random nodes distribution, but also it can be used as a criterion to recognize the neighboring nodes in the workspace.

According to the introduced TDG, each TDG cell includes nodes that their end-effector positions are placed in that cell. Thus, all the nodes that are placed in one TDG cell are considered as cellmates as well as node neighbors; then the local planner is called to connect the cellmates and generate a linear trajectory between them. Moreover, all the nodes of eight surrounding cells are considered as the node neighbors (Figure 3).

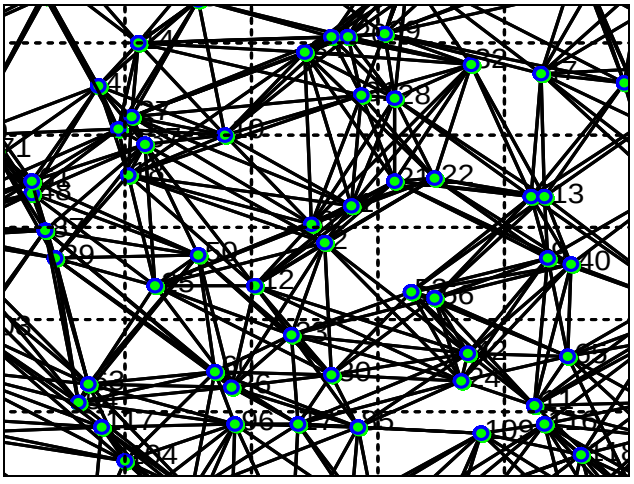


Figure 3. 7-dof manipulator node neighbors selection; workspace dimension= 10 unit by 10 units, density= 1

8) Local Planner

The local planner is in charge of connecting the neighbor nodes. The straight-line segments in Figure 3 show the connections between two nodes. However, the trajectory between two neighbor nodes, which is not necessarily a straight line, is generated by the local planner while satisfying the collision free condition.

The quality of the collision detection for the trajectories depends on the workspace and obstacles' geometry. For instance, if there are obstacles with sharp edges and vertices the local planner must generate paths with higher resolution.

9) Cost Function

Path planning for highly redundant manipulators involves cost functions that have to be optimized since there are large numbers of degrees of freedom performing a task. Moreover, there are different criteria and conditions for the robotic arms since they are designed to perform various types of tasks in different workspaces. Hereby, one of the main contributions of the proposed technique for hyper-redundant robotic arms motion planning, is handling and optimization of various cost functions simultaneously. Here, a simple and common cost function is defined which can be applicable in industrial tasks that optimize multiple cost functions simultaneously:

- Distance: finding the shortest Euclidean distance between the end-effector positions from initial to goal configuration.
- Time: finding the shortest trajectory between initial and goal configuration.
- Total joints displacement: the objective is having minimum joints displacement in accordance to the assigned costs for each link.

In most cases, the manipulator must rapidly perform a task in the shortest path while the cumulative links movements are minimized.

2.2. The Query Phase (real-time)

By using the constructed roadmap in the off-line preprocessing phase, paths are to be found between any initial and goal configurations during the query phase. In another word, once the roadmap is generated, it contains occupancy grid, all valid nodes, and all feasible obstacle-avoiding trajectories between neighbor nodes and initial robot properties. The strategy that has been used in the query phase for connecting any arbitrary pairs of nodes is finding the shortest/least cost path between any desired start and goal nodes that involves the graph searching algorithm A*. The workflow of the algorithm is shown in Figure 4.

A* is the best-first graph search algorithm that has been widely used in path finding because of its performance and accuracy to find the efficient path. Additionally, other studies found A* to be superior to other methods [11]. Detailed information about A* can be found in [12] and [13].

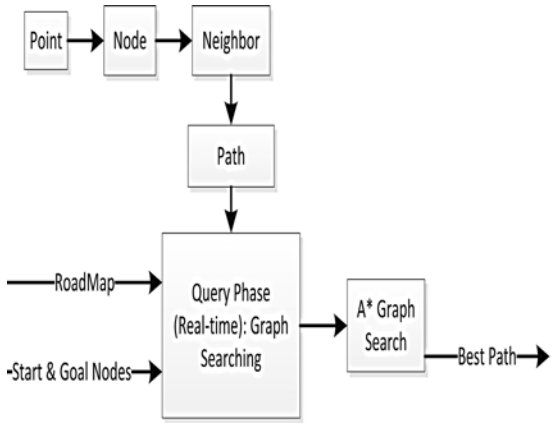


Figure 4. Real-time query phase workflow

3 Implementation

The proposed approach is implemented in MATLAB platform because this platform is widely accepted for academic purposes as well as in industry. The overall procedure of implementation of proposed algorithm can be listed as follows:

- Robot and links initialization (D-H parameters)
- Environment and obstacles initialization
- Tip distribution grid initialization
- Generation of random configurations
- Collision detection algorithm
- Recognition of neighbor nodes
- Local Planner
- A* graph search algorithm

The workflow of the developed modified PRM planner is depicted in Figure 5. According to this workflow, the off-line phase starts with generation of obstacle-avoiding random configurations followed by the robot and environment initialization to create a roadmap to begin the query phase, which needs a pair of initial and goal nodes to find the best path.

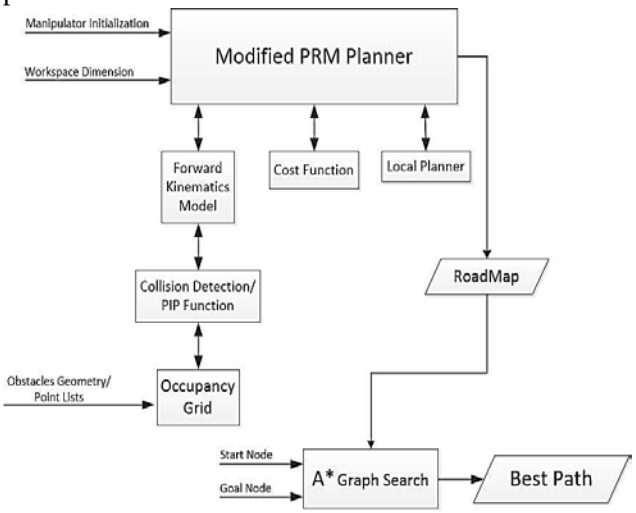


Figure 5. Overview of presented planning algorithm.

To efficiently develop and implement the proposed approach in MATLAB, one should use the advantages of object-oriented programming OOP since the planner is dealing with large number of nodes and numerous properties for each node as well as the edges and the trajectories between each pair of neighbor nodes. Figure 5, demonstrates the structure overview of this planner including the classes, relations and connections. Thus, one may define a class for each type of data that is like a template for the creation of specific instances of that class. The instances or objects contains actual data for a particular entity that is represented by the class. In addition, objects are not just passive data containers. Objects actively manage the contained data by allowing certain functions to be executed. By precluding data that does not need to be public for other classes and preventing external clients from access and manipulation of local data, all class functions can only be executed for a specific object and their public access are denied. Using this structure enables the planner to perform wide range of variety of user-defined problems.

4 Result and Discussion

Implementation and all the simulations have been done in MATLAB framework using Windows 8.1 on a Lenovo ThinkPad W530 processing with Intel Core i7-3820QM 2.70 GHz (64-bit) processor and 8 GB of RAM.

In the results figures, the manipulator and its joints are shown in bold black line, and green circles, respectively. The obstacles are shown in cyan line segments.

4.1. Scenario No.1

In the first scenario, the developed framework simulates a model of 7-dof Canadarm 2 [14], to pick up an object form an initial location and move it along the free space and finally place it in a tight location surrounded by a box-like obstacle. Figure 6 presents one of tasks that the Canadarm2 can encounter while operating at the International Space Station (ISS); which is simply done by the proposed planner. The final configuration is exactly the best approached configuration to place the end-effector deep inside a narrow space. The 109-node roadmap is generated in 276.4 seconds and the A* algorithm finds the best path in 34.8 seconds. The density of roadmap is 2 with the resolution of 60 units per axis for occupancy grid mapping.

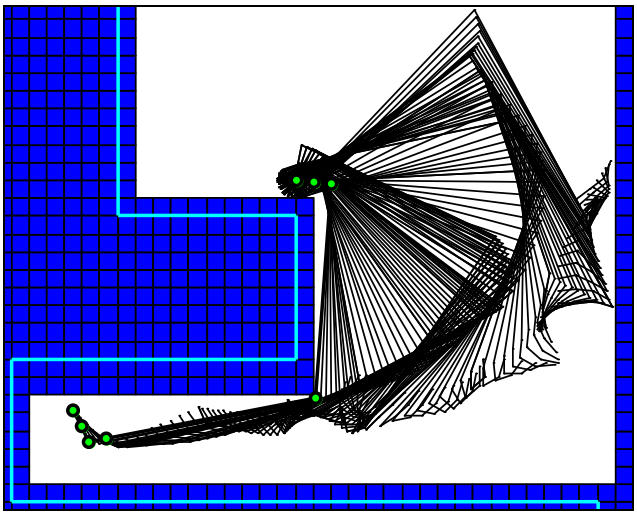


Figure 6. The designed trajectory for Canadarm2 in scenario No. 1.

4.3. Scenario No.2

In this scenario the performance and capability of the proposed motion planner is checked for an RRP manipulator having a prismatic joint. Figure 7 shows one of the superior features of the proposed method which has not been addressed properly in other works in the literature.

Scenario No. 2 is designed to illustrate the advantage of using prismatic joint for an end-effector for reaching the farthest boundaries of a workspace. The 46-node roadmap is generated in 39.3 seconds and the A* found the best path in 2.7 seconds. The density of the roadmap is 1 with the resolution of 60 units per axis for occupancy grid mapping.

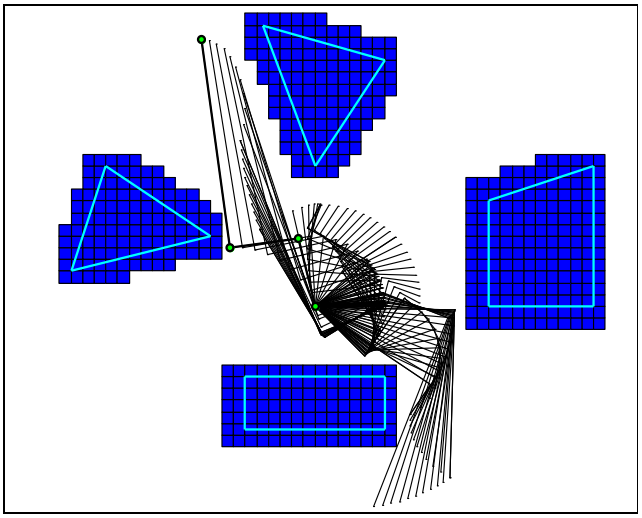


Figure 7. The designed trajectory for a RRP manipulator in scenario No. 2.

4.5. Scenario No.3

Last scenario presents the capability of the proposed planner for generating a collision-free trajectory of a 16-dof hyper-redundant manipulator which has not addressed comprehensively in other studies in the literature. Sharp-edge obstacle avoidance of a robotic arm with high number of DOF is one of the most important features of the proposed planner.

Figure 8 shows a defined a scenario for a 16-dof manipulator avoiding collisions in a highly constrained workspaces with sharp-edge obstacles.

The 150-node roadmap is generated in nine minutes and the A* found the best path in 40.3 seconds. The density of roadmap is 4 with the resolution of 100 units per axis for occupancy grid mapping. Obviously this scenario takes longer in off-line stage, having said that the planner deals with a hyper-redundant manipulator as well as a high-resolution occupancy grid as a result of special features of the obstacles in this scenario.

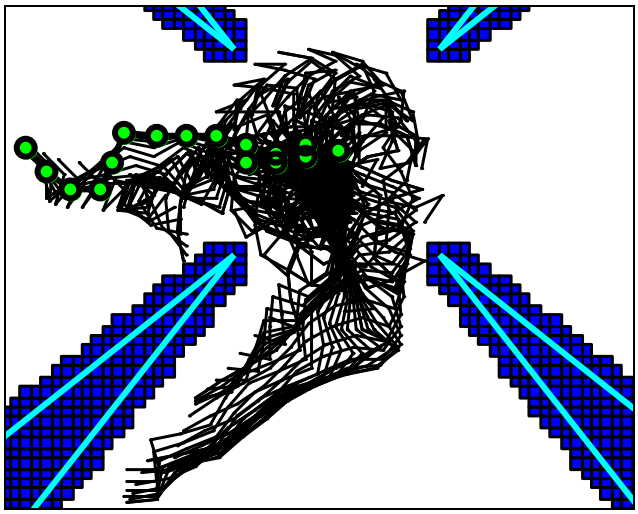


Figure 8. The designed trajectory for a 16-DOF manipulator.

Table 1 shows the performance of the proposed trajectory planner for selected scenarios. Its rows represent degrees of freedom of the tested manipulators, total generated collision-free nodes, density, occupancy grid resolution, off-line CPU-time, Real-time CPU-time, C_{obs}/C_{free} area and Longest Link/Narrowest Channel.

According to this table, due to the length of roadmap trajectories, density and occupancy grid resolution, the CPU-time for off-line phase increases up to nine minutes. However, the real-time phase is done a quite short period of computation time for a 16-dof robotic arm in comparison to other methods and non-sampling based approaches.

Table 1. Comparison between the results of defined scenarios

	Scenario 1	Scenario 2	Scenario 3
Degrees of Freedom (DOF)	7	3 (RRP)	16
Density	2.0	1.0	4.0
Occupancy Grid Resolution	60	60	100
Total Number of Nodes	109	46	150
Off-line CPU-time (sec)	276.4	39.31	539.3
Real-time CPU-time (sec)	34.80	2.70	40.30
C_{obs}/C_{free} Area	0.635	0.155	0.138
Longest Link / Narrowest Channel	3.000	1.834	0.150

5 Conclusions

A comprehensive efficient collision-free trajectory planner for hyper-redundant manipulators has been developed based on a multiple-query PRM approach. The developed general motion planner is capable of handling all required steps for path planning as well as all types of high-dimensional manipulators, different cost functions operating in constrained environments with various sets of obstacles in 2D. It is important to notice 2D path planning is more constrained, so more difficult, than 3D path planning and that the proposed planner is capable of implementing 3D collision avoidance. Hyper redundant manipulators with a very large number of DOF have been successfully simulated in highly constrained workspaces, by the proposed fast and efficient planner. The developed algorithm can be expanded by dynamically-optimized motion planning or energy-optimized collision-free trajectory.

ACKNOWLEDGEMENTS

The authors also would like to express their appreciation to Mr. Mehrdad Bakhtiari for his valuable suggestions and advice on the path planner programming.

This paper research has been supported by a grant (No: 155147-2013) from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References:

[1] T. Lozano-Perez, "A simple motion-planning algorithm for general robot manipulators," Robotics and Automation, IEEE Journal of, 3(3), pp. 224-238, 1987.

[2] I. D. Walker, "Kinematically redundant manipulators," In Springer Handbook of Robotics, pp. 245-268, 2008.

[3] A. Feizollahi and R. V. Mayorga, “On the modeling and the optimal motion planning of manipulators via a modified D* Lite search algorithm”, WSEAS Transactions on Systems and Control, vol. 12, pp. 148-157, 2017.

[4] J. H. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robot," Robotics and Autonomous Systems, vol. 27, no. 3, pp. 171-194, 1999.

[5] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato, "A machine learning approach for feature-sensitive motion planning," In Algorithmic Foundations of Robotics VI, pp. 361-376, 2005.

[6] R. Bohlin and E. E. Kavraki, "Path planning using lazy PRM," in ICRA'00. IEEE International Conference on Vol. 1, 2000, pp. 521-528.

[7] G. Song, S. Miller, and N. M. Amato, "Customizing PRM roadmaps at query time," in Proceedings 2001 ICRA. IEEE International Conference on Vol. 2, 2001, pp. 1500-1505.

[8] S. M. LaValle, Planning algorithms. Cambridge university press, 2006.

[9] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," Robotics and Automation, IEEE Transactions on, 12(4), pp. 566-580, 1996.

[10] P. Sabetian, Feizollahi, C. F. A., and S. A. A. Moosavian, "A compound robotic hand with two under-actuated fingers and a continuous finger," in In Safety, Security, and Rescue Robotics (SSRR), Kyoto, 2011, pp. 238-244.

[11] W. Zeng and R. L. Church, "Finding shortest paths on real road networks: the case for A*," International Journal of Geographical Information Science, pp. 531-543, 2009.

[12] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," Systems Science and Cybernetics, pp. 100-107, 1968.

[13] P. E. Hart, N. J. Nilsson, and B. Raphael, " A formal basis for the heuristic determination of minimum cost paths," Systems Science and Cybernetics, vol. 4, no. (2), pp. 100-107, 1968.

[14] R. V. Mayorga, F. Janabi-sharifi, and A. K. W ong, "A Fast Approach For The Robust Trajectory Planning Of Redundant Manipulators," Journal of Robotic Systems, vol. 12, no. 2, pp. 147-161, Feb. 1995.