Implementation of a New Backtrack Free Path Planning Algorithm for Manipulators

Md. Nazrul Islam

Supervised by

Professor Shinsuke Tamura

A thesis submitted in partial fulfillment of
the degree of Doctor of Engineering at
the Program of System Design Engineering
Graduate School of Engineering, University of Fukui, JAPAN

September 5, 2008

Acknowledgments

Firstly all praises belong to almighty Allah.

I would like to express my deep profound gratitude and respect to my supervisor Prof. Shinsuke Tamura for his direction, support, supervision, patience, and boundless enthusiasm throughout all of my research. Heartfelt thanks to him for his kind guidance, which has been proved invaluable in completion of the thesis. Concrete research ideas are provided by him and also with his direct guidance the ideas came into reality.

I am truly and enormously grateful to my collaboration research supervisor Prof. Jaime Gallardo-Alvardo, Department of Mechanical Engineering, Instituto Tecnolo`gico de Celaya, Me`xico, for my collaboration research about "kinematics and dynamics of parallel manipulators". There are no words to express how grateful I am for his unlimited generosity. Not to forget: Prof. Jaime Gallardo-Alvardo, thanks for all generous assistance and so on.

I would like to thank to Dr. Tatsuro Yanase and Dr. Syuji Taniguchi for their co-operations during my study in this Lab. I am thankful to all of under graduate and postgraduate students of the laboratory for their social, academic and administrative support and friendship. I would also like to thank all of the staff at university of Fukui. My special thanks go to Prof. Kiyoshi Nakashima, International Student Center.

I am indebted to my brothers, sisters and all family members for their moral support and patience in making this piece of work successful.

I would like to greatly acknowledges the patience, sacrifice, and affection of my beloved parents whose love, support and encouragement helped me to come to this present state. Many thanks to my parents for always believing in me and for always giving me full moral support no matter the situation. The last, the last is the most precious, I am infinitely thankful to my beloved wife, Tinni, for being there even when being far away, for all her understanding, support and love always being there for me.

Md. Nazrul Islam

September 5, 2008

Dedication

I dedicate this work to my parents. I also dedicate this work to my wife, Tinni.

Abstract

In this research, a new backtrack free path-planning algorithm (BFA) for multi-arm manipulators that calculates paths by searching grid points in Euclidean space directly instead of Configuration space is implemented. Currently available resolution complete path-planning algorithms cannot be applied to manipulators with many arms, because their computation time and memory space for calculating collision free paths increase exponentially with the number of arms. Here, it is assumed that positions in the workspace of manipulators are approximated by finite number of grid points, and a resolution complete algorithm is the one that can determine the existence of paths and find correct paths if they exist, when grid sizes are small enough. Therefore usual planners adopt heuristics that are not adequate for automated and real time applications, i.e. sometimes they cannot find paths even they exist, and it is not possible to estimate path calculation times in advance.

A newly proposed backtrack free path planning algorithm (BFA) solves this problem. BFA is an exact algorithm, i.e. it is backtrack free and resolution complete. Different from existing resolution complete algorithms, its computation time and memory space are proportional to the number of arms. Therefore paths can be calculated within practical and predetermined time even for manipulators with many arms, and it becomes possible to operate multi-arm manipulators in complicated and fully automated environments.

This thesis describes the implementation and evaluation results of BFA. BFA was implemented for the path planning in *2*-dimensional environments and evaluated while changing the number of arms and obstacle placements. Its performance under locus and attitude constraints was also evaluated. In all of the experiments, collision free paths were found within less than few seconds. The computation volume of the algorithm is almost the same as the theoretical one even for complicated cases. Namely, BFA calculates paths with much shorter time than existing algorithms with constant performance independent of environments. Also BFA enables easy locus and attitude constrained path calculations.

Table Of Contents

Chapter Title **Page**

List of Figures

List of Tables

Chapter 1

Introduction

A manipulator is a mechanism in which a sequence of arms are connected by joints i.e. fulcrums, which are located between neighboring arms. It changes its attitudes by varying angles of these joints to handle and convey things. Path planning is a process to find paths that bring start attitudes of manipulators to their goal attitudes while avoiding collision with obstacles, and it is one of the most important tasks for operating manipulators. The requirement for the automatic planning is growing in high variety low volume productions, where production environment such as machine layout changes more frequently than traditional low variety high volume productions, i.e. every change of the environment requires path planning. Efficient algorithms for solving problems of this type have important applications also in areas such as medical, space, painting, etc. However, currently available resolution complete path-planning algorithms cannot be applied to manipulators with many arms, because their computation time and memory space for calculating collision free paths increase exponentially with the number of arms [21, 50]. Here, positions in the workspace of manipulators are approximated by finite number of grid points and a resolution complete algorithm is the one that can determine the existence of paths and find correct paths if they exist, when grid sizes are small enough. Therefore usual planners adopt heuristics that are not adequate for automated and real time applications, i.e. sometimes they cannot find paths even they exist, and it is not possible to estimate path calculation times in advance. To enable real-time path planning of manipulators with many arms, this research implements and evaluates a new backtrack free path planning algorithm (BFA) that was proposed in [77].

Many approaches to path planning had been proposed already until now. They can be classified into three categories, road map [1-3, 14-17, 19-21, 55-57, 66-69], cell decomposition [34] and potential field approaches [23-25]. Here almost all of these algorithms find collision-free paths in the Configuration space (C-space), because an attitude of a manipulator can be represented as a single point in C-space; an attitude of an N arms manipulator can be represented by a set of angles of N individual joints. Algorithms based on roadmap-based approach, find collision free paths by iterating two steps, i.e. in the first step, a sequence of sub-goals toward the goal attitude are defined, and in the second step, paths that connect adjacent sub-goals are calculated. In this iteration sub-goals are

changed when paths that connect adjacent subgoals cannot be found. The cell decomposition based approach finds paths by connecting cells (blocks of the workspace of the manipulator) that are not occupied by obstacles, while iteratively subdividing them when there is no such cell, and in the potential field based approach, artificial potential functions are defined so that collision free paths can be found by tracing them. However all of these algorithms are heuristics based, and their efficiency is limited when free spaces have complicated shapes.

BFA removes these difficulties by finding paths in Euclidean space directly for individual arms sequentially instead of C-space. BFA is resolution complete and its computation time and memory space are the linear order of the number of arms, i.e. computation time and memory space required for the algorithm are the order of NMR, provided that M and N are the number of grid points included in the work area of the manipulator and the number of arms, respectively. R is the maximum number of grid points on the surfaces of spheres constituted by the moving ranges of individual arms. Moreover because BFA is backtrack free, it can find paths with almost the constant performance independent of the complexity of the workspace, e.g. different from heuristics based approaches it can determine non-existence of paths promptly. BFA is also applicable to path planning problems with locus or attitude constraints. It can find constrained paths without reducing its performance.

This research discusses the implementation and performance issues of BFA in complicated environments. BFA was implemented for path planning in 2-dimensional work space, and computation times and path lengths were measured. In all evaluation scenarios with complicated environments, BFA showed substantially better performance than existing algorithms. Also different from existing algorithms, BFA performance is not sensitive to environment, i.e. it performs independent of environments [81, 85].

In Chap. 2, several basic terms are introduced with related works. In Chap. 3, BFA is described. Chapter 4 evaluates BFA under various environments and Chap. 5 summarizes the work.

Chapter 2

Backgrounds

This chapter, introduces several basic terms with related works. In Section 1, basic terms, manipulator, work space, configuration space (C-space) and path planning are introduced. Related works are described in section 2 and 3. Section 4 concludes this chapter.

2.1 Basic Terms

2.1.1 Exact and Heuristic Algorithms

There are two kinds of path planning algorithm in terms of their completeness: i.e. exact algorithms and heuristic algorithms. Advantages and disadvantages exist in both types of algorithms. A path planning algorithm that is guaranteed to find a solution if one exists, and report failure if there is no solution, is said to be complete, and exact algorithms are complete. The resolution completeness is related to discretization. When continuous quantities such as obstacle dimensions or configuration parameters are discretized, the associated algorithm is inherently approximate. However, its accuracy can be arbitrarily improved by increasing the resolution of discretization. If an algorithm is exact in the limit as the discretization approaches a continuum, it is called resolution complete. The complete algorithms are usually computationally expensive. They cannot be applied to manipulator with many arms, because their computation volume for calculating collision free paths increases exponentially with the number of arms.

The complexity of complete path planning algorithms lead researchers to seek heuristic methods with weaker notions of completeness, such as probabilistic completeness [34, 42]. An algorithm is probabilistically complete if its probability of finding a path (if one exists) can be increased to 1 with the computation time, i.e. if paths can be found when infinite time is allowed. The heuristic based algorithms find solutions without examining all possibilities, therefore paths are not guaranteed to de found even they exist. They may fail to find solutions for difficult problems, or they try to find solutions forever when path does not exist. Heuristic algorithms are fast in many cases but they include backtracks and it is not possible to estimate calculation times in advance.

2.1.2 Manipulators

A manipulator is a mechanism in which a sequence of arms are connected by joints i.e. fulcrums, which are located between neighboring arms, and it can changes its attitudes by varying angles of these joints to handle and convey things. They are one of the most important tools in factories. They are used for welding, assembly and other manufacturing processes. Figure 2.1 shows a typical example of a manipulator. This manipulator contains three arms and a gripper. Fulcrums of individual arms are called their joints and other end positions are called their movable ends and these joints enable individual arms to move. DOF (Degree of Freedom) is the number of independent movements that generated by these arms. Then, the manipulator in the Figure has *6* DOF. In the remainder of this thesis, it is assumed that a manipulator consists of *N* arms, and successive integers (i.e. *1*, *2*, ---, *N*) are assigned to them so that the base arm becomes the *1*st one. To simplify discussions, it is also assumed that the joint position of the 1st arm in Euclidean space is fixed, and the movable end of the *n*-th arm coincides with the joint of the (*n*+*1*)-th arm.

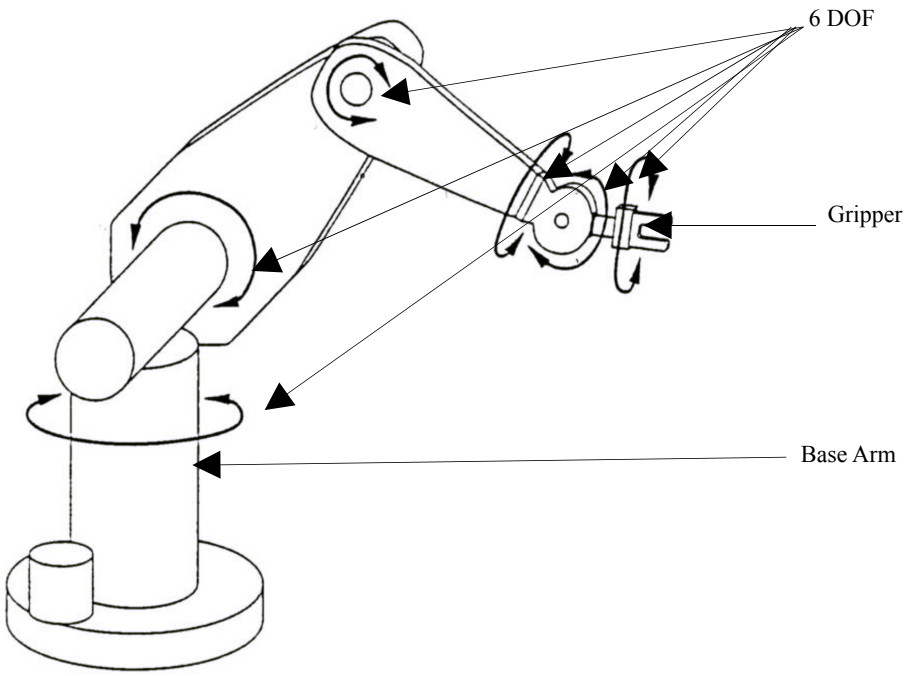


Figure 2.1 Six Degrees of Freedom (DOF) Multi-Arms Manipulator

2.1.3 Work Space

Work space is the *2*-D or *3*-D Euclidean space in which manipulator and obstacles are located. Positions in the work space can be represented by (x, y) coordinates in *2*-D case and (x, y, z) coordinates in *3*-D case. The configuration of a manipulator can be represented by a set of position coordinates of joints of individual arms. Positions in the work space are approximated by finite number of points. In this thesis, it is assumed that this approximation is accomplished by points that constitute a grid.

2.1.4 Configuration Space (C-Space)

The configuration space plays very important roles in path planning. In almost all path planning algorithms, manipulator configurations (attitudes of a set of individual arms) are represented in the configuration space (C-Space). The configuration of an object of a given shape is a single point in the multi-dimensional space that corresponds to a set of independent parameters that characterizes the attitude of the object. Six parameters are needed to specify the configuration of a rigid body in three dimensional work space (*3* for the position, *3* for the orientation). For a manipulator consists of *N* arms, the configuration can be specified by the angles of *N* joints. Figure 2.2 (a) shows that a configuration of a *2*-arms manipulator can be represented by the joint angles $\theta 1$ and $\theta 2$ (other choices are possible). Therefore the number of degrees of freedom is *2* in this case. Figure 2.2 (b) is a collision-free motion of a *2*-arms manipulator between the start and the goal configurations S and G among polygonal obstacles, and Figure 2.2 (c) is the corresponding path representation in the C-Space. The shaded regions correspond to the configurations of obstacles. As shown in these Figures C-Space enables the simple representation of manipulator movements, i.e. a movement of *2*-arms manipulator from the start to goal attitudes in the workspace can be represented as a simple single curve in the C-space.

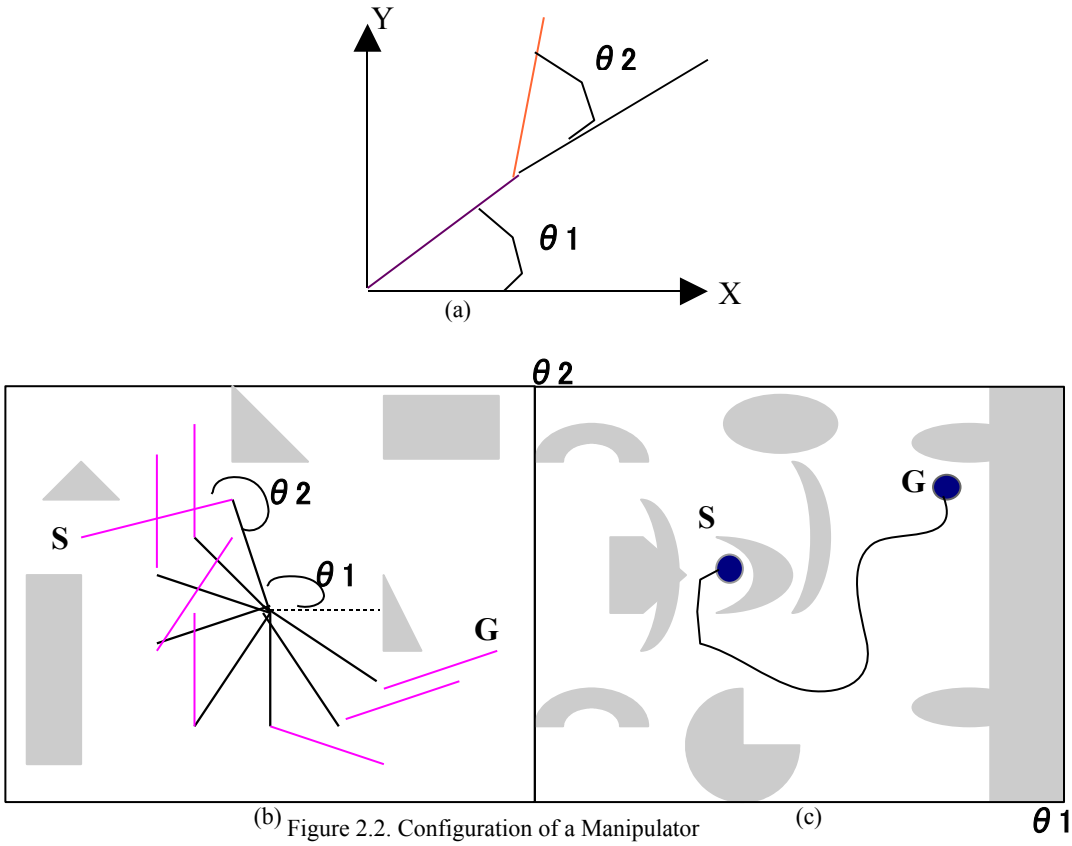


Figure 2.2. Configuration of a Manipulator

2.1.5 Path Planning

The path planning is a process to find paths that bring start attitudes of manipulators to their goal attitudes. The main concerns of path planning are to move arms from the start to the goal configuration, to avoid collision with obstacles, to avoid collision with other arms and to move arms efficiently.

Figure 2.3 shows an example of the path planning process. This is two degrees of freedom manipulator. Here S is the start attitude and G is the goal attitude. There are several obstacles around the manipulator. Arms must move from S to G while avoiding collision with any obstacle if paths exist.

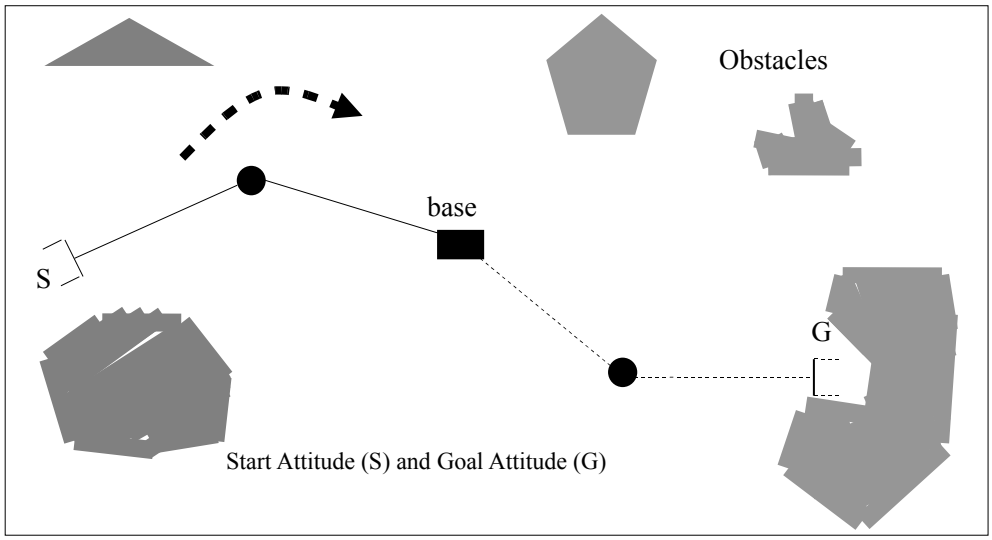


Figure 2.3 Path Planning Process

2.2 Related Works

This section offers a survey of the recent advances in path planning algorithms for manipulators. It is divided into 2 parts, i.e. classification of path planning algorithms and descriptions of existing algorithms.

2.2.1 Classification of Path Planning Algorithms

Existing path planning algorithms can be classified into three categories, road map [5, 8, 26, 30, 32, 41, 45, 49, 51, 52, 60, 61,73, 79], cell decomposition [34, 65, 83], and potential field approaches [11, 27, 40, 44, 48]. However all of these algorithms are heuristics based, and their efficiency is limited when free spaces have complicated shapes. Here almost all of these algorithms find collision-free paths in the C-space, because an attitude of a manipulator can be represented as a single point in C-space.

2.2.2 Roadmap

The roadmap approach represents the free-space for a manipulator as a collection of connected collision free paths. This sets of collection free paths is called a roadmap. A path is generated as

follow, firstly a path from the start attitude to some part of the roadmap and that from the goal attitude to the roadmap are constructed. Then, by using standard graph algorithms, the roadmap is searched for a path between the start and the goal attitudes. Here a roadmap is a graph of which nodes are points sampled from the free space F, a collision-free subset in a given configuration space C, and edges are simple collision free paths, e.g. straight-line segments, that connect these nodes. Because this approach is heuristics based, it requires frequent backtracks, i.e. when sampled nodes cannot be connected by collision free paths to points in the roadmap, a roadmap is replaced by a newly sampled one. For a static environment, the roadmap is constructed once, and can be used to solve multiple planning problems. This approach is most efficient when a potential field method is used as the local operator.

The many variations of the roadmap approach differ mainly in the method for constructing the roadmap. These variations include: visibility graph [4], Voronoi diagrams [18], freeways networks [31], and randomized roadmaps [62, 63]. The first three are among some of the earlier attempts to build path planners, and they are applicable only for simple mobile robots with two or three degrees of freedom. However the recent roadmap approach in which the roadmap is constructed using randomized techniques, have been found experimentally to capture the structure of a manipulator free space in a efficient manner, even for complex manipulators with many degrees of freedom. In a series of papers, L. E. Kavraki and J. C. Latombe [28], L. E. Svestka et. al. [42], and L. Kavraki [31] laid the ground for probabilistic roadmap (PRM) methods. PRM works in two phases: a learning phase, and a query phase. In the learning phase, the configuration space is randomly sampled for collision-free configurations. These configurations form the vertices in a graph , i.e. a roadmap. A simple local planner is used to look for connections between nearby vertices. If a connection is found between configurations, an edge connecting the corresponding vertices is added to the graph. In the query phase, the start configuration and the goal configuration are connected by a sequence of sampled configurations that was constructed in the learning phase, then the path planning problem has been reduced to a graph search problem, which can be answered without long computation time. Dijkstra's shortest path algorithm is often used to find the best path.

The idea of PRM is that the roadmap will eventually give a sufficient representation of free configuration space (the set of all feasible manipulator configurations in the C-space) when randomly sampled vertices are added repeatedly in the learning phase. Figure 2.4 (a) is an example of a roadmap for a two-dimensional configuration space and (b) is an example of a roadmap query. The resulting path is shown by the thicker lines.

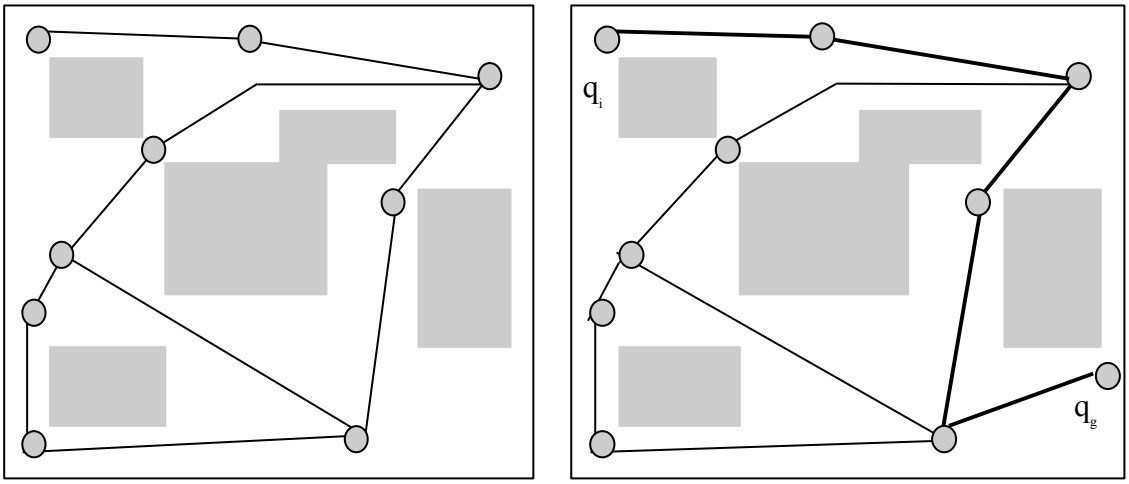


Figure 2.4: (a) a Roadmap for *2*-dimensional C-Space. (b) a Roadmap Query

PRM are particularly useful if repeated queries are expected for the same environment, because the cost for constructing the roadmap is amortized over each query. PRM planners have for a long time been thought of as multiple query planners due to the costly learning phase, but recent contributions by R. Bohlin and L. E. Kavraki [58, 62] have changed on that; they showed that the costly operation of verifying whether path segments are collision free could be postponed until the query phase. In this scheme, the constructed roadmap contains many infeasible edges, and they are detected and deleted during the query phase. By this approach, the number of required collision detections were significantly reduced and the approach becomes competitive as a single shot planner as well.

2.2.3 Potential Field

The potential field method proposed by O. Khatib [6] is a popular approach for implementing real-time path planning. It enables relatively simple and efficient path planning and have taken an important place. The basic idea is to consider the manipulator to be moving in a field of forces. The overall potential field is made up of an attractive field, which attracts the manipulator towards its goal attitudes, and a repulsive field that pushes it away from the obstacles in the environment. Namely, in the potential field approach, a scalar potential function that has high values near obstacles and the global minimum at the goal is constructed, and the manipulator moves in the direction of the negative gradient of the potential. An important drawback of potential fields is that

the manipulator tends to get trapped in local minima, when the potential is not designed deliberately. These minima in the potential surface prevent the manipulator from reaching its destination.

The potential fields have been refined for a number of years and used for many applications including path planning for manipulators. They offer a simple yet efficient method to encode the location of obstacles in a given environment through a representation that can be directly interpreted by classical path planning techniques. In subgoal-graph approach, subgoals are set as key configurations that are useful for finding collision free paths. A graph of subgoals is generated and maintained by a global planner, and a simple local planner is used to determine the reachability among subgoals. This two-level planning approach has turned out to be most effective path planning method [10]. However these refinements still make use of simple heuristic algorithms, which may yield a quick result but do not guarantee the finding of paths.

2.2.4 Cell Decomposition

The most common approach to path planning is based on a cell decomposition of a C-space. A cell is a region of the free-space with a simple shape such that a path can be easily constructed between any two configurations within the cell. By describing the free-space as a collection of cells, path planning can be reduced to a search of the graph representing the adjacency relationship between two cells. A collision free path between the start and goal configurations of the manipulator is found by first identifying the two cells containing the start and the goal attitudes and then connecting them with a sequence of connected cells. Cell decompositions can be exact or approximate.

Exact decompositions use cells that can precisely represent the free-space; therefore cells must be described by complex analytical expressions. Planners based on exact cell decompositions tend to be more of theoretical interest as they are complex to implement and extremely inefficient. On the other hand, such planners can prove the existence of paths exactly and find paths when they exist.

Approximate decompositions use some simple cell shape, typically a rectangular, to represent the free-space up to a given resolution. The regular shape of the cells results in simplified algorithms for generating and representing the decomposition. A planner that uses an approximate cell decomposition may fail to find a path even one exists, however, such failure occurs only when the manipulator must move through a region of the free-space that is smaller than the resolution of the decomposition. Namely, approximate cell decomposition is not complete, but can yield similar, if not exactly the same, results as exact cell decomposition. However, the trade-off for this accuracy is a more difficult mathematical process.

The major limitation of cell decomposition planners is that the number of the cells tends to grow exponentially with the dimension of the configuration space. This property limits the planners to manipulators with no greater than perhaps four degrees of freedom.

2.3 Existing Algorithms

Despite of the fact that many researchers had proposed different path planning approaches for multi-arm manipulators, the process is more complex and the computation time explodes largely when number of arms is greater than six. Therefore the majority of classical path planning techniques cannot be directly transposed to multi-arm manipulator, and consequently various heuristics are introduced. This chapter briefly describe some of heuristics used in existing algorithms.

The PRM [82] is a road map based algorithm and currently considered as one of the most efficient path planning method. Figure 2.5 shows the behavior of the PRM. PRM builds a roadmap E by sampling up to the pre-defined number of configuration points from free space F. In Figure 2.5, this pre-defined number is set to S, i.e. E consist of S sampled points. In the Figure, two functions FreeCon(q) and FreePath(q, q') that calculate logical values true or false are used. For any point $q \in C$, FreeCon(q) is true if and only if $q \in F$; and for any point pair q, $q' \in C$, FreePath(q, q') is true if and only if q and q' can be connected with a straight-line segment lying entirely in F. Where C represents the whole C-Space.

The performance of PRM depends on favorable "visibility" properties; where two points in the C-space are called visible with each other if they can be connected by a collision-free straight-line segment. When this condition is satisfied, PRM can find paths quickly. It can find paths by searching only a little part of C. However when the above condition is not satisfied, PRM cannot find paths efficiently. The poor visibility property of F is caused by narrow corridors for example as shown in Figure 2.6. In the Figure, obstacles are represented by black areas, therefore free space F consists of two big white areas connected by a corridor. For PRM, it is difficult to find a path that connects points q1 and q2 when the width of the corridor is small. Because PRM is heuristics based, it cannot give answers when there is no path that cannot connect q1 and q2.

```
1.  if freepath(q1, q2) is true then return the straight
    path  from q1 to q2.
2.  else initialize the roadmap E with two nodes, q1 and
    q2.
3.  repeat
4.   random sample a configuration q from C uniformly
     at random.
5.   if FreeConf(q) is true then add q as a new nodes of
      E.
6.   for every node v of E such that v ≠ q do
7.     if FreePath(q, v) is true then add (q, v) as a new
        edge of E
8. until q1 and q2 are in the same connected  component
    of E or E contains S+2 nodes.
9.  if q1 and q2 are in the same connected component of
      E then
10.    return a path between q1 and q2
11. else
12.    return NoPath
```

Figure 2.5 Overall Structure of this PRM Frameworks

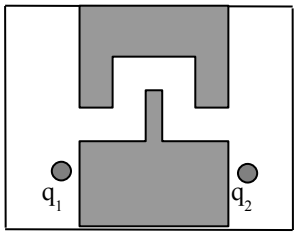


Figure 2.6 An Example of Difficult Cases for PRM

M. Saha and J. C. Latombe proposed a new method "small-step retraction" to find paths through such passages [80]. The "Small-step retraction" method is significantly faster (sometimes by several orders of magnitude) than pre-existing planners. This method consists of 3 parts, i.e. slightly "fattening" of manipulator's free space, construction of a roadmap in fattened free space, and repairing of the roadmap through retraction of fattened space. Fattened free space is not explicitly computed, instead, the geometric models of workspace objects (manipulator arms and obstacles) are "thinned" around their medial axis. Two repair strategies are proposed, i.e. optimist and pessimist. Optimist assumes that generated paths traverse true passages, and it postpones repairs as much as possible. Then after having found a complete path in F^* between the start and the goal attitudes, it tries to repair this path by retracting the colliding portions into F. Here, F^* denotes the fattened free space. If the path cannot be quickly repaired, it simply returns failure. On the other hand, Pessimist immediately repairs every configuration sampled in $\partial^* F$, instead of waiting until a path in F^* has

been found. Here, $\partial^* F$ is the fattened boundary. The former is usually very fast, but may fail in some pathological cases. The latter is more reliable, but not as fast. A simple combination of the two strategies yields an integrated planner that is both fast and reliable.

Z. Sun et. al. presented a hybrid sampling strategy in the PRM framework for finding paths through narrow passages [79]. When the environment includes narrow passages, to capture the visibility of F in the roadmap, it is essential to sample points in these narrow passages. This, however, is difficult, because of their small volumes. Any volume-based sampling distribution is likely to fail. In particular, the uniform distribution does not work well. A key idea of the method is to reduce sample density in parts with high visibility, resulting in increased sample density in narrow passages. The method can be implemented efficiently in high-dimensional configuration spaces using only simple tests of local geometry.

In the Cell decomposition based algorithm proposed by A. Hourtash and M. Tarokh, a manipulator is decomposed into several chains [65]. Where a chain is a combination of several consecutive links, and paths for individual chains are planned independently to be combined to construct the path for the whole manipulator. The algorithm consists of off-line and on-line methods. The off-line method generates a set of attitudes for each chain that do not collide with obstacles as collision-free discretized attitudes, and the on-line method finds a sequence of discretized attitudes that include the start and goal attitudes. Two major disadvantage of this planner are firstly the computation time increases exponentially with number of chains and secondly a complicated link path that may consist of twists and turns are generated. Of course the algorithm is heuristics based and backtracks slows the planner.

The sequential search strategy [13, 33, 40, 50] reduces the computation times of the potential fields method. The essence of this approach is to exploit the serial structure of manipulator links and decompose the n-dimensional problem for a n links manipulator into a sequence of smaller m-dimensional sub-problems, each of which corresponds to the motion of m links, i.e. it calculates motions of individual links sequentially based on the paths calculated for the previous links. The other idea is to define a number of discrete control points, then numerical potential field is defined for each of these control points. It has better performance for low DOF (up to 8) problems with small number of backtracks, however its success rate is not good with many backtracks for more than 8 DOF manipulators.

T. Nishimura et. al., used potential fields in conjunction with genetic algorithms [35]. The potential fields are used to guide the end effector, while the genetic algorithm ensures collision avoidance for the rest of the structure of the manipulator. The variations in joint angles are coded as genes, and

these genes are then passed through a series of genetic operations: fitness, crossover, mutation, natural selection and parameter tuning. These operations generate a pseudo-random attitudes to be evaluated. There are a few drawbacks; most notably is that the algorithm finds a solution through lengthly pseudo-random means. S. Pires and T. Machado also proposed an approach, in which a penalty function that represents the configuration of manipulator (obstacles, manipulator position, angular speeds,…) is minimized by using the fundamental operations of genetic algorithms [53].

G. Oriolo et al. proposed a heuristic-like approach where the given end-effector path is followed in a tracking operation [70]. In the tracking operation the path is segmented into smaller steps, and every possible configuration of the manipulator is analyzed until one is found that does not collide with the environment. Those solutions are found in a random order and do not guarantee the optimality in terms of joint displacement and computation time.

C. Lin and J. Chuang offered a different perspective on manipulator path planning [74]. They proposed to use guide planes (GP) as intermediate goals in the 3D workspace. Using continuous repulsive fields, the algorithm finds the path with the lowest value for repulsion within the boundaries of the GP. Although this method yields good results, it is necessary to give that GPs in advance. Also when obstacles are located closely, it cannot work correctly. G. Lian et al. proposed a simple approach [71] in which repulsive forces are calculated by neural networks and fuzzy logic to push the manipulator away from obstacles. However the example provided is too simple compared with real applications. In the algorithm proposed by S. Ando, a general path is found using global path planning methods in which sub-goals are found throughout the global path by using a general A* algorithm [75]. The approach aims to reduce the computation times. However, the strategy does not seem to encompass general manipulator architectures, e.g. no consideration is made on the inverse kinematics. In fact, the results presented only deal with the path of the end effector.

J. Barraquand and J. Latombe have proposed a classical approach [12] in which paths are generated while following the gradient of potential-field. If the manipulator becomes stuck in a local minimum, the algorithm tries to escape by the addition of random movements. The proposed approach shows good results in dealing with manipulators with a large number of DOF, however, the approach is very lengthy when dealing with narrow corridors. The random search for a valid solution leads to non-uniform planning times and is not repetitive; this represents a major drawback for real-time applications. M. Park et al. proposed a similar approach [64] where once the manipulator is trapped in a local minimum, a random solution is found using simulated annealing from the set of neighbors to the current solution.

S. Caselli et al. presented a method for escaping local minima by using multiple escaping strategies [63, 72]. The first method (Straight Line) is to move the manipulator in a random "up-hill" direction until a criterion is met. The second method (Straight Line Select) eliminates undesirable candidate directions therefore optimizing the escaping path and minimizing the occurrence of the manipulator falling in the same minima. Although the method is simple, the results obtained are not suitable for real-time applications.

H. Chang proposed to apply different forces to different parts of the manipulator [43]. The trajectories found by this algorithm demonstrate that the approach is able to fully model the manipulator, however, the algorithm requires significant amount of memory and lengthy computation times. L. Chengqing et al. presented method to escape local minima [59]. Local minima are created most of the time by concave objects, or a series of objects forming together a concave object in the workspace. Therefore in the method, when the manipulator is trapped in local minima, it tries to find the largest opening of the concave object. However, it seems not adequate for real-time applications.

A limiting factor of the potential field approach is that generation of the potential field is cumbersome especially for evolving environments. M. Piaggio and A. Sgorbissa proposed a method to statistically reduce the calculation [54]. Their approach is to divide an area near to the manipulator or end effector into circular sectors of equal width and into equally spaced rings. The resulting grid will have a similar shape to that of a cylindrical coordinate system. The sector explored is updated beginning with the smallest radius where an object is found. The approach has shown a reduction of 11% in computation time over traditional grids. In the approach proposed by Y. Kitamura [36] the quadtree representation is used. Although the path planning algorithm consists only of heuristic techniques, the quadtree approach shows considerable improvement over the regular grid-like representation. The approach reduces the number of nodes to explore in order to determine the best path. The method is also applicable to mobile robot-type manipulator since rotations are also calculated.

E. Conkur and R. Buckingham examined highly redundant manipulators and used them in very crowded environments [46]. In order to speed up the process, objects are modeled by simple analytic ones, i.e. obstacles are modeled as ellipses with a security margin and the arms of the manipulators are modeled as lines. Since the manipulator consists of highly redundant arms, the interaction between the arms also be taken into consideration.

T. Laliberté has proposed method that reduces the occurrence of local minima [29]. The potential fields are discretized and the attractive field is computed by means of wave propagation from the

target position. It has been validated on redundant 2D and 3D manipulator arms, but demonstrates limitations when the manipulators try to reach behind obstacles. The proposed approach relies on the analytical solution to the inverse kinematics of the manipulator, which increases complexity and limits the generality of the solution.

A hierarchical collision free path planning algorithm proposed by W. K. Hyun and H. Suh [37], consists of two parts; tunnel finding and path planning. The tunnel finding algorithm constructs a free subspace that includes start and goal configurations and a collision free path candidates can be found. The tunnel is constructed by using big cells defined as a group of several basic cells. The path planning algorithm then plans a path in the tunnel. The purposes of the algorithm are (i) a restriction of searching space to reduce computational burden, (ii) removal of undesired zig-zag sectors of the path which are produced when there is a cell lumping, and (iii) recovery of failures in planning a path. It is remarked that this hierarchical collision free path planning algorithm utilizes only local information such as distance between neighboring cells; it does not require either a large memory size to store information on whole work space and excessive computational time. However, the algorithm is efficient only for 4 DOF scara type manipulator.

N. Kawarazaki and K. Taguchi proposed a method consists of two steps [38]. First, a free form surface is defined that covers collision free regions and includes start and goal points in the configuration space, and a collision free Path-Restricted-Curved-Surface (PRCS) is generated. The PRCS is described by Bezier surface. Second step is to generate the optimized path on the PRCS. The path on the PRCS is selected to construct a geodesic line that connects from start to goal points. The geodesic line in the configuration space is the most suitable path in the point to minimize the total value of manipulator's joint angle changes. The algorithm is suitable for 3-6 DOF manipulators but it is difficult to create the most suitable collision-free path, even though information about the manipulator and obstacles are all known.

M. Tarokh proposed a fast path planning algorithm by formation-posture decomposition [39]. The algorithm consists of an off-line phase followed by an on-line phase. In the off-line phase certain defined or respecified body formations and arm postures are generated and collision of the manipulator at these defined formations and postures with obstacles are checked and stored in the form of a collision table. This off-line phase is carried out only once for a particular manipulator and workspace environment. In the on-line phase, a search is carried out to find a collision free sequence of adjacent body formations and arm postures. As a result of this formation-posture decomposition and separation of computation into off-line and on-line phases, the algorithm is able to achieve short on-line path planning times of few seconds for typical industrial manipulators working in reasonably

cluttered environments. Path planning is essentially performed in the work space thus avoiding the costly mapping of obstacles into the C-space. The path planner is fast, and on-line planning time is a few seconds for a modified Puma 560 working in environment containing some twenty complex shaped obstacles.

V. Moreno, E. Sanz and F. J. Blanco proposed an approach based on graph search techniques in C-space [47]. It is based on a temporal parameterization of the state variables. There are many measures to study the algorithm performance, but two of them are more significant: the trajectory length and the generated nodes, i.e. the execution time. The numerical simulation only for industrial manipulators are considered therefore all the concepts are restricted to these ones. A parallel graph search algorithm has been developed with the aim of carrying out the planning with a low computational cost and it can be easily used for dynamical path planning tasks. However, due to the high computational cost involved in the problem, it is little difficult to use for on-line applications.

2.4 Conclusion

Almost all path planning algorithms of manipulators find collision-free paths in the C-space. The reason is that the attitudes of multiple arms of a manipulator can be represented as a single point in C-space, and it brings systematic ways for finding paths of the manipulator. However for manipulators that have many arms, the dimension of C-space becomes too large to develop resolution complete paths planning algorithms that calculate paths within practical time. Therefore it is not practical yet to apply them to manipulators with many arms. Although various heuristics including random algorithms such as genetic ones are used so as to avoid exhaustive searches of possible paths, they are not free from backtracks and cannot ensure the finding of correct paths even they exist, and consequently it is difficult to establish fully automated path-planning processes by existing algorithms.

Chapter 3

A New Backtrack Free Path Planning Algorithm

This chapter describes backtrack free path planning algorithm (BFA) [77], that is implemented and evaluated in this research. BFA is back track free and resolution complete. Different from existing resolution complete algorithms, its computation time and memory space are proportional to the number of arms. Therefore paths can be calculated within practical time even for manipulators with many arms, and it becomes possible to apply it to manipulators that operate in complicated and fully automated environments.

In section 3.2 overview of the algorithm and in section 3.3 definition of terms are described. In section 3.4 some assumptions, and in section 3.5 and 3.6 the basic theorem and the algorithm are described. Section 3.6 discusses the computation volume, and in section 3.7, conclusion of this chapter is presented.

3.1 Introduction

BFA finds collision free paths by searching grid points in Euclidean space directly instead of C-Space for individual arms sequentially. The algorithm is resolution complete and backtrack free under the assumption that arms themselves can collide with each other. Here, the work area of the manipulator is approximated by finite number of grid points. Computation time and memory space required for the algorithm are the order of NMR, provided that M and N are the number of grid points included in the work area of the manipulator in Euclidean space and the number of arms, respectively. R is the maximum number of grid points on the surfaces of spheres constituted by the moving ranges of individual arms. Moreover because BFA is backtrack free, it can find paths with almost the constant performance independent of the complexity of workspace, e.g. different from heuristics based approaches it can determine non-existence of paths promptly. BFA is also applicable to path planning problems with locus or attitude constraints. It can find constrained paths without reducing its performance. Regarding to the assumption, collisions of arms themselves can be easily removed through local adjustments of paths because usually there are enough free spaces arround collision free attitudes.

3.2 Overview of the Algorithm

Figure 3.1 shows the overall structure of the algorithm consists of off-line and real-time parts. The off-line part is executed only when locations of the manipulator or obstacles are changed. Firstly, it calculates R(*N*), a set of grid points in *2* or *3*-dimensional Euclidean space, to which the movable end of the *N*-th arm can reach from its initial position. Here, *N* is the maximum arm number, and the movable end of the *N*-th arm is considered as a single point that is not constrained by other arms: therefore simple algorithms can be used for calculating R(*N*). Then, for each n beginning from $n = N$ to *1*, it finds feasible attitude sets of the *n*-th arm at individual points, and based on them, it calculates R(*n-1*), a set of grid points to which the joint of the *n*-th arm can reach from its initial position without collision by connecting mutually (*n-1*)-connecting points (described in later section).

```
/* off-line part */
  calculate R(N), a set of grid points to which the movable end of
    the N-th arm can reach from the start position as a single point
  n=N
  while (n > 0) {
      find feasible attitude set A(X, n) of the n-th arm at
          each point X in the workspace
      determines the (n-1)-connectivity of individual neighboring
          point pairs
      calculate R(n-1), a set of points, which are reachable by the
          joint of the n-th arm from its start position, based on
     (n-1)-connectivity
      n=n-1
  }
  /* real-time part */
 if ( Dn∈R(n), and Dn∈A(Dn-1, n) for all n) {
      n=1
      while (n=<N) {
          find the locus of the movable end of the n-th arm
          that connects its start position to its goal position
        n=n+1
    }
  }
  else {there is no collision free path}
```

Figure 3.1 Overall Structure of the Algorithm

Here, feasible attitude set A(X, *n*) of the *n*-th arm at point X is a set of grid points that can be occupied without collision by its movable end, when its joint is located at X. It must be noticed that A(X, *n*) is calculated based on feasible attitude sets of the (*n+1*)-th arm; therefore for each feasible

attitude A of the *n*-th arm, there exist at least one collision free attitudes of the *m*-th arm that are consistent with A for all m ($n<m\leq N$). The real-time part is executed every time when the goal attitude is given to the manipulator, and based on R(*n*), a locus of the movable end of the *n*-th arm that brings its initial position H_n to its goal position D_n, is calculated for each *n*, starting from *n = 1* to *N*. As discussed later, R(*n-1*) represents the points that are reachable by the joint of the n-th arm from the initial position without collision as a set of n-th, (*n+1*)-th, ---, and *N*-th arms, therefore, existence of paths is ensured when $D_n \in R(n)$ and $D_n \in A(D_{n-1}, n)$ are satisfied for all *n* at the beginning of the real-time part, and loci of individual arms can be determined sequentially from *n = 1* to *N*, without any backtrack. Here, $D_n \in A(D_{n-1}, n)$ means that the attitude of the *n*-th arm, of which joint and movable end are located at their goal positions, is feasible.

3.3 Definition of Terms

Location and attitude of an arm: A location of the *n*-th arm is represented by the grid point occupied by its joint. An attitude of the *n*-th arm is represented by a pair of grid points (X, Y). Here, X and Y are grid points occupied by the joint and the movable end of the *n*-th arm, respectively.

Feasible attitude set (FAS): Attitude (X, Y) of the *N*-th arm (*N* is the maximum arm number) is called feasible when the *N*-th arm does not collide with any obstacle. Also attitude (X, Y) of the n-th arm ($n < N$) is called feasible when the *n*-th arm does not collide with any obstacle and there exists at least one feasible attitude (Y, Z) of the (*n+1*)-th arm. A feasible attitude set (FAS) of the *n*-th arm at X is a set of grid points that are occupied by the movable end of feasible attitudes of the *n*-th arm located at X and denoted as A(X, *n*).

Successive attitudes: A pair of attitudes of the n-th arm (X_1, Y_1) and (X_2, Y_2) is called successive, when X_1 and Y_1 are equal or adjacent to X_2 and Y_2, respectively.

Connecting point pair: Any grid point pair P and Q included in A(X, *n*) and A(Y, *n*) is called a connecting point pair of a FAS pair A(X, *n*) and A(Y, *n*), when attitudes of the *n*-th arm (X, P) and (Y, Q) are successive.

***n*-connectivity:** Adjacent grid points X_1 and X_2 (X_1 is considered to be adjacent to X_1 itself) are called *N*-connective, when they are not occupied by any obstacle. Adjacent grid points X_1 and X_2 are

called (*n-1*)-connective, when a FAS pair A(X_1, *n*) and A(X_2, *n*) has at least one connecting point pair (Y_1, Y_2) such that Y_1 and Y_2 are *n*-connective.

***n*-reachable set:** *n*-reachable set R(*n*) is a set of grid points that are reachable from H_n, the initial position of the movable end of the *n*-th arm, by chaining grid points, which are mutually *n*-connective.

3.4 Assumptions

A1. Continuity of the space :
An arm can move its attitude from (X_1, Y_1) to (X_2, Y_2) without colliding with obstacles as a single arm, when attitudes (X_1, Y_1) and (X_2, Y_2) are collision free and successive.

A2. Self-collision free arms
Collisions among arms of the manipulator are allowed.

Assumption A1 can be satisfied always when the intervals of adjacent grid points are small enough compared with the sizes of arms and obstacles: therefore A1 does not limit the applicability of the algorithm. Concerning A2, although arms cannot collide with each other in actual applications, it is not a serious constraint either collisions among arms can be removed easily by local adjustments of their positions, because usually, especially in *3*-d cases, there are enough free spaces around collision free attitudes. Namely, collisions can be removed by moving relevant joints around their current positions as shown in Figure 3.2. In the Figure, attitudes (P, Q) and (R, S) of the n-th and (*n+2*)-th arms collide at position Z. However because there exist collision free attitudes in the vicinity of (P, Q), (Q, R) and (R, S), this collision can be removed by changing arm attitudes to (P, $\underline{Q}$), ($\underline{Q}$, $\underline{R}$) and ($\underline{R}$, S), respectively.

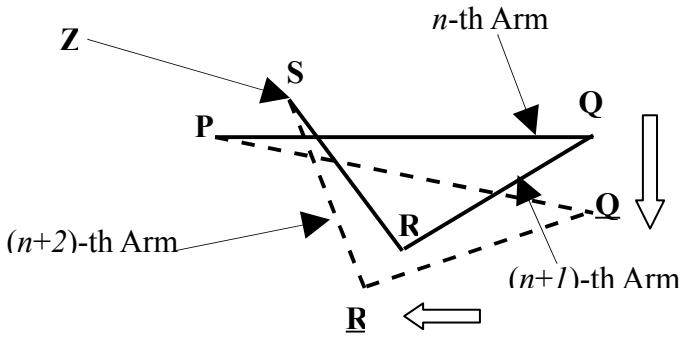


Figure 3.2 Avoidance of Collisions Among Arms

In addition to A1 and A2 the following conditions are also assumed, to make discussions comprehensive (these assumptions will be removed later).

A3. Connectivity of FAS

FAS A(X, *n*) of the *n*-th arm at position X is a connected set of grid points in terms of *n*-connectivity for every X and *n*.

A4. Uniqueness of FAS

Elements of A(X, *n*) can be uniquely determined independent of locations of the (*n-1*)-th arm. Namely, the *n*-th arm at X can move its movable end to same positions even the (*n-1*)-th arm changes its attitude.

Assumption A3 can be satisfied when no obstacle exists in the work area of the manipulator. Assumption A3 means that 2 different feasible attitudes of the *n*-th arm at the same position can be mutually reachable without collision. Figure 3.3 shows the case where A3 is not satisfied. Because of an obstacle that has a hole in it, FAS of the arm at position P is divided into two disjoint connected parts, i.e. A(P, *n*) is not *n*-connective. Apparently this kind of situation does not happen when no obstacle exists.

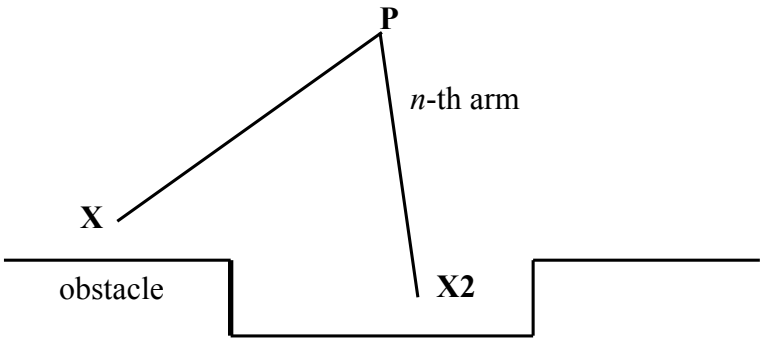


Figure 3.3 Disjoint Connected Sets in FAS

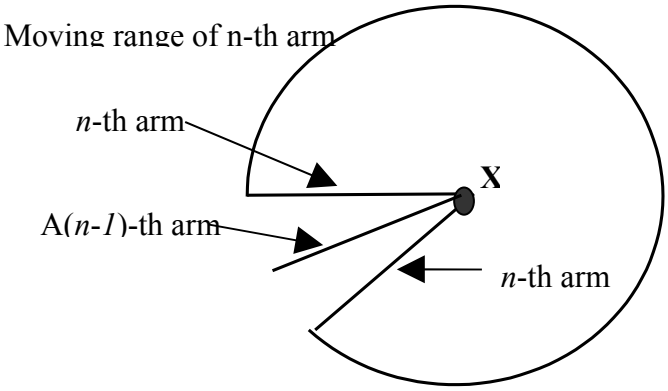


Figure 3.4 Moving Range of an Arm

A4 is also satisfied usually. In usual cases, the moving range of the n-th arm does not constitute the whole surface of the sphere so as to avoid collisions between the *n*-th and the (*n-1*)-th arm as shown in Figure 3.4, and consequently, A(X, *n*) may vary depend on attitudes of the (*n-1*)-th arm. However in many cases, the moving range of the *n*-th arm constitutes almost the whole surface of the sphere; therefore even when the algorithm generates path L, in which the *n*-th arm takes an attitude that overlaps with that of the (*n-1*)-th arm, collision free paths that have the same effect as L can be found in the vicinity of L by the local position adjustments as same as in Figure 3.2.

3.5 Basic Theorem

Then under assumptions A1-A4, it can be proved that an arbitrary grid point P in *n*-reachable set R(*n*) can be reached by the movable end of the *n*-th arm from its initial position while avoiding collision with obstacles, provided that P is included in A(X, *n*) for some point X which is reachable

by the movable end of the (*n-1*)-th arm from its initial position. Therefore a backtrack free algorithm can be constructed, i.e. when the locus of the movable end of the n-th arm from its initial position H_n to its goal position D_n is calculated and $A(D_n, n+1)$ includes the goal position of the movable end of the (*n+1*)-th arm D_{n+1}, it is ensured that the locus of the (*n+1*)-th arm that brings its movable end from H_{n+1} to D_{n+1} while avoiding collision with obstacles also exists. More precisely, the following theorem can be proved. In the theorem, H and D, initial and goal attitudes of the manipulator, are represented as sets of initial and goal positions of movable ends of individual arms, i.e. H= $\{H_0, H_1, H_2, ----, H_N\}$ and D = $\{D_0 = H_0, D_1, D_2, ----, D_N\}$.

[Basic Theorem] Under the assumptions from A1 to A4, the necessary and sufficient condition for the existence of collision free paths of a manipulator from its initial attitude H = $\{H_0, H_1, H_2, ----, H_N\}$ to the goal attitude D= $\{D_0 = H_0, D_1, D_2, ----, D_N\}$ is that n-reachable set R(*n*) and $A(D_{n-1}, n)$ includes D_n for each *n* (= *1*, *2*, ---, *N*). Also when collision free paths from H to D exist, L_0, the locus of the joint of the 1st arm is the point $\{H_0\}$, and L_n the locus of the movable end of the *n*-th arm can be determined without backtracks based on L_{n-1}.

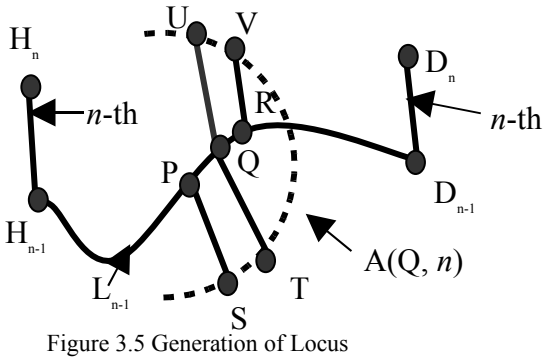


Figure 3.5 Generation of Locus

The below is the outline of the proof. Apparently L_0, the locus of the joint position of the *1*st arm exists, i.e. it consists of a single point $\{H_0 = D_0\}$. Therefore, let us assume that L_{n-1}, a sequence of mutually (*n-1*)-connective points from H_{n-1} to D_{n-1} can be constructed, and P, Q and R are consecutive points on L_{n-1}. Then because P and Q, and Q and R are (*n-1*)-connective, there exist pairs of feasible attitudes [{P, S}, {Q, T}] and [{Q, U}, {R, V}] of the *n*-th arm as shown in Figure 3.5. Moreover, because A (Q, *n*) is connected in terms of n-connectivity, A (Q, *n*) includes a sequence of points that connects T to U, and consecutive points in the sequence are mutually n-connective. By applying this

process to every point in L_{n-1}, it is possible to constitute a sequence of mutually *n*-connective points, i.e. locus L_n of the movable end of the *n*-th arm from H_n to D_n.

3.6 The Algorithm

It is straightforward to constitute a backtrack free path planning algorithm from the above proof procedure. Figure 3.6 describes the algorithm. In the algorithm, a locus of the movable end of the *n*-th arm is calculated as an *1*-dimensional array L_n. A joint position of the *n*-th arm corresponding to the movable end position $L_n(j)$ is represented as $SL_n(j)$.

(Off-line part)

Initialization {

Calculate F(*n*) for each n (*n = 1* to *N*). Here, F(*n*) is a set of grid points that the movable end of the *n*-th arm can reach as a single point when no obstacle exists. Then remove grid points that are occupied by obstacles from F(*n*).

Calculate *N*-connectivity table C(*N*), by making neighboring grid point pairs X_1 and X_2 in F(*N*) mutually *N*-connective.

}

R(n) generation {

n=N

While (*n* > 0) {

Calculate *n*-reachable set table R(*n*) by chaining mutually n-connective grid points in C(*n*) from the initial position of the movable end of the n-th arm H_n. Calculate A(X, *n*) for each grid point X in F(*n-1*). Then for each neighboring grid point pair X_1 and X_2 in F(*n-1*),

if (A(X_1, *n*), A(X_2, *n*) include connecting point pair Z_1, $Z_2 \in$ R(*n*)) and Z_1 and Z_2 are *n*-connective){

Make X_1 and X_2 mutually (*n-1*)-connective, and register {Z_1, Z_2} as connecting point pair of A(X_1, *n*) and A(X_2, *n*) in C(*n-1*).

}

n=n-1

}

}

(Real-time part)

Path generation {

if ((some R(*n*) does not include D_n) U (some A(D_{n-l}, *n*) does not include D_n) {

Quit /* path does not exist */

}

else {

$L_0(0) = H_0$ /* locus of the joint of *1*st arm */

leng=*l* /* length of the locus */

n=1

While (*n* =< *N*) {

$L_n(0)$ = Current = H_n $SL_n(0) = H_{n-1}$

/* initialize locus of the *n*-th arm */

prev= leng

/* length of the locus of (*n-1*)-th arm */

leng = *1* /* length of the locus of *n*-th arm */

j=l

While (j =< prev){

If (j =prev) {/* end of (*n-1*)-th arm locus */

Joint*1* = D_n

}

else {

Joint*1* = Z_j

Joint*2* = Z_{j+1}

/* pair Z_j and Z_{j+1} is a connecting point

pair of A(L_{n-1}(j), *n*) and A(L_{n-1}(j+ *1*), *n*) */

}

Move the movable end of *n*-th arm at L_{n-l}(j) from Current to

Jointl, by connecting mutually *n*-connective grid points in

A(L_{n-1}(j), *n*). Let {P_1, ---, P_m} be the obtained sequence,

```
        then, store {P1, ---, Pm} to the area from Ln(leng) to
        Ln(leng+m-1). Also store Ln-1(j) to the area from SLn(leng) to SLn(leng+m-1).
                                leng = leng + m
                                Current = Joint1

                                If (j≠prev) {
                                        Ln(leng) = Joint2
                                        SLn(leng) = Ln-1(j+1)
                                        Current = Joint2
                                        leng = leng +1
                                }
                                j=j+1
                        }
                n=n+1
            }
        }
    }
```

Figure 3.6 BFA

Backtrack free feature of BFA enables path calculations with computation time and memory space proportional to the number of arms as discussed in Sec. 3.7; therefore, it becomes possible to apply it to manipulators with many arms that operate in complicated environments. However, serious situations happen about assumption A3, when a manipulator must move through areas surrounded by multiple obstacles as shown in Figure 3.7. Figure 3.7 shows the case to rotate the movable end of the n-th arm at A, from X_1 to X_2. In this case, because the n-th arm collides with an obstacle when it rotates around A, A(A, *n*) has multiple connected components, and theorem is not applicable. However, even path-planning problems for these cases can be converted to the one, in which assumption A3 is satisfied, i.e. all arms have FASs with single connected components at each grid point therefore assumption A3 can be removed.

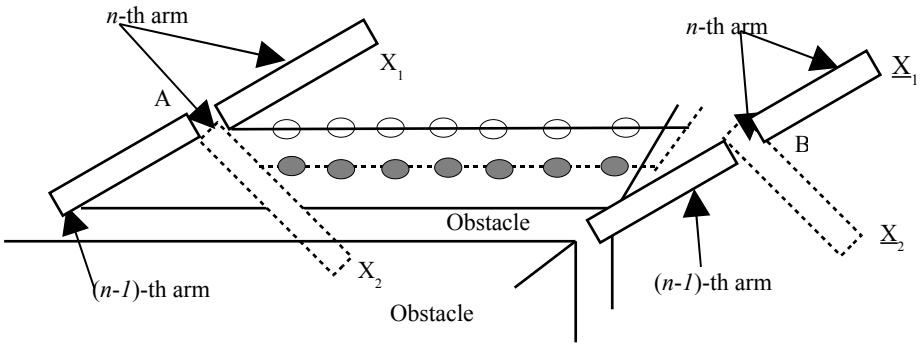


Figure 3.7 Path Surrounded by Obstacles

The conversion is done as follows, namely, when A(X, *n*) has multiple connected component A_1, A_2, ---A_m, copies X_1, X_2, ---X_m of gird point X that occupy the same position as X are created in F(*n-1*) (here, F(n) is the work area on which movable end of the n-th arm is located), and A(X_j, *n*) is defined as A_j. Then, the converted problem satisfies assumptions A1-A4. Also the algorithm does not generate paths that connect X_j to X_k directly, because these copies are not mutually n-connective (A(X_j, *n*) and A(X_k, *n*) are disjoint sets). Therefore, Theorem is applicable to the converted problem that includes multiple points at same positions, and paths that avoid collision with obstacles can be calculated even for cases shown in Figure 3.7. In the Figure, two copies (they are represented as white and black circles) are automatically generated for individual positions on the line between A and B in F(*n-1*). White and black copies correspond to connected components of FASs that include points in area X_1 and X_2 as the movable end positions of the n-th arm, respectively. Then, because white and black copies are not connective, the (*n-1*)-th arm moves from the white copy to the black copy of A (i.e. changes its attitudes from (A, X_1) to (A, X_2)), while tracing white copies until B, where the *n*-th arm has a single connected FAS and white and black copies merge, to change the attitude from (B, $\underline{X}_1$) to (B, $\underline{X}_2$), and tracing black copies back to A. Here, copy creation processes are trivial; they only require n-connectivity checking of neighboring grid points in FASs.

Here, it must be noticed that different copies X_j and X_k of the same grid point X in F(*n-1*) are not mutually (*n-1*)-connective. Therefore, when A(Z, *n-1*) includes X and does not include a sequence of mutually (*n-1*)-connective grid points from X_j to X_k, A(Z, *n-1)* has disjoint *(n-1)*-connective sets that includes X_j and X_k, respectively. Then grid point Z also has multiple copies in F(*n-2*); namely, copies of grid points in F(*n*) introduce copies also in F(j) for $j < n$. In the example shown in Figure 3.8, A(P, *n+2*) has *4* disjoint (*n+2*)-connective sets corresponding to areas A, B, C and D, therefore grid point

P has *4* copies in F(*n+1*). Then, A(Q, *n+1)* has 5 disjoint (*n+1*)-connective sets that correspond to area E and 4 copies of P in area D, and this means that Q has *5* copies in F(*n*). In this way, copies generated in higher arm analysis propagate to lower arm analyses, and increase computation complexity of the algorithm. However in many cases, this propagation terminates, because usually there is an arm position with a FAS that includes points connecting these copies. In the Figure, A(R, *n*) includes V, from which the movable end of the *n*-th arm can be moved to every copy of Q, because the (*n+1*)-th and (*n+2*)-th arms can possess any kind of attitudes at V. Then A(R, *n*) is a connective set that includes all copies of Q, and therefore, *R* does not need to have its copy. Even when all arms have *G* copies at all grid points and the copy propagation does not terminate, total number of copies can be limited to $G^N M$, and the computation volume is still much less than existing resolution complete algorithms. In actual applications, the computation time and memory space does not increase so much, because many grid points do not have copies.

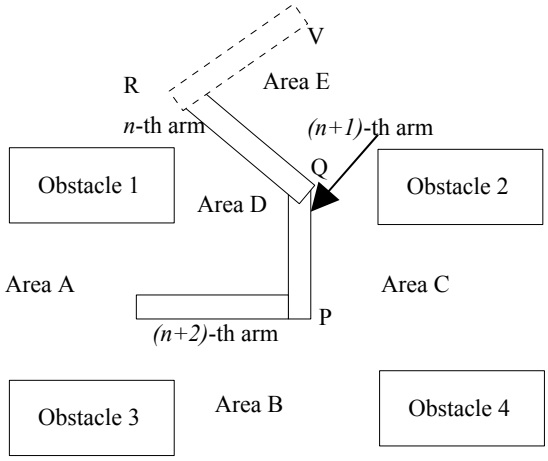


Figure 3.8 Disjoint FASs of Arms

Regarding to assumption A4, in actual cases, the moving range of an arm does not constitute a surface of the whole sphere as shown in Figure 3.4, and FAS of the *n*-th arm at X changes depending on the location of the (*n-1*)-th arm. Namely, A(X, *n*| Q_1) and A(X, *n*| Q_2) may differ when Q_1 and Q_2 are different. Here A(X, *n*| Q) is a feasible attitude set of the *n*-th arm at X provided that the (*n-1*)-th arm is located at Q. The basic Theorem is also applicable to these cases, by making copies of X in F(*n-1*) corresponding to locations of the joint of the (*n-1*)-th arm. However, although problems can be converted to ones that satisfy A4, it is not practical, because every grid point X in F(*n-1*) has many copies corresponding to the number of feasible attitudes of the (*n-1*)-th arm that have X as their movable end position. This difficulty can be removed by the fact that *2* copies X_1 and X_2 of X

in F(n) can be merged into a single copy, under the condition that X_1 and X_2 are n-connective with the same copies of points, which are neighboring to X. This condition is satisfied when the moving range of the n-th arm is not too small. Especially, copies of X corresponding to A(X, n|Q_1) and A(X, n| Q_2) in F(n-1) can be merged into a single copy, when A(X, n| Q_1) and A(X, n| Q_2) cover more than half of the sphere surfaces centered at X, and this is satisfied by the most of manipulators. When moving ranges of individual arms cover almost the whole sphere surfaces, local adjustment of paths is enough as shown in Figure 3.2, i.e. it is not necessary to create copies.

3.7 Computation Volume

Computation time and memory space required for the algorithm execution is the order of *NMR* as described below. Here, *M* and *R* represents the total number of grid points in the work area and the maximum number of grid points included in individual FASs, respectively. *N* is the number of arms. Primary calculations required in the algorithm are those for calculating A(X, n) and R(n). A(X, n) calculation is the process of checking collisions between an arm and obstacles. For a given attitude of an arm, collisions can be checked with the computation time proportional to the arm length, there are at most *R* different attitudes for each joint position of the arm, and there are *M* different joint positions and *N* different arms. Therefore the computation time required for calculating all A(X, n) is the order of *NMR*. For R(n) calculation, firstly, n-connectivity between every adjacent point pair X and Y must be checked, and this is achieved by searching a connecting point pair {P, Q} in A(X, n+1) and A(Y, n+1). Then, because A(X, n+1) includes at most R different grid points, and Y and Q are neighboring grid points of X and P respectively, the total computation time for this checking is the order of *NMR*. R(n) itself can be calculated by the algorithm of Dijkstra. In this case, there are N arms, the number of grid points is less than M, and individual grid points have fixed number of neighboring grid points that is proportional to the dimension of the space; therefore all R(n) can be calculated with the computation time of the order of *NM*.

The largest part of the memory space required is that for maintaining A(X, n) for individual arms and their locations. Because there are *N* arms and *M* different grid points, and individual A(X, n) has *R* grid points at most, the total memory space is the order of *NMR*. However, the algorithm does not require all A(X, n) simultaneously; therefore the total memory space actually required for the algorithm execution can be reduced to the order of *NM*.

3.8 Conclusion

A resolution complete path planning algorithm for multi-arm manipulators that searches Euclidean space directly is introduced. This algorithm is backtrack free under the assumption that arms of the manipulator can collide with each other, and required computation time and memory space can be reduced to the order of NMR. Here, M and N are the number of grid points that cover the work area of the manipulator, and the number of arms, respectively. R is the upper bound of the number of grid points on surfaces of spheres constituted by the moving ranges of individual arms. Collisions among arms themselves derived from the assumption can be removed from paths easily by their local adjustments.

Chapter 4

Evaluation of BFA

In this chapter, the performance of BFA is evaluated for *2*-dimensional environments while changing the number of arms and obstacle placements [81, 85]. Its performance under locus and attitude constraints is also evaluated. Evaluation results show that the computation volume of the algorithm is almost the same as the theoretical one, i.e. it increases linearly with the number of arms even for complicated cases. Moreover BFA achieves the constant performance independent of environments.
In section 4.2 test cases and in section 4.3 evaluation results are described. In section 4.4 conclusion of this chapter is presented.

4.1. Introduction

BFA is resolution complete and its computation time and memory space are the linear order of the number of arms. Moreover because it is backtrack free, it can find paths with almost the constant performance independent of the complexity of workspace, e.g. different from heuristics based approaches it can determine non-existence of paths promptly. BFA is also applicable to path planning problems with locus or attitude constraints. It can find constrained paths without reducing its performance.
In this chapter the performance of BFA was tested and evaluated while changing obstacle placements and the number of arms. The off-line part computation time, the real-time part computation time, the total computation time, the total number of copy points and the maximum number of copy points generated in path calculations, and the total path length are measured. Here, the maximum number of copy points means the largest number of copy points that are generated for a single point, and the total path length means the sum of path length that are passed by all arms. In all evaluation cases, manipulators work in *4m x 4m* square area in *2*-dimensional space and bases of the manipulators are located at the center of the environments. The area is divided into *80* x *80* grid points, i.e. the length of the grid interval is *5 cm,* and the total number of grid points becomes to *80* x *80* x *N,* when the manipulator has *N* arms. The algorithm were implemented by Java and executed on Windows XP running on *1.53* GHz CPU with *224* M Bytes of RAM. In all evaluation scenarios with complicated

obstacle placements, BFA showed significantly better performance than existing algorithms.

4.2 Test Cases

Figures 4.1 - 4.5 show the obstacle placements used in the evaluations. They also show the start and goal attitudes of arms. In each Figure, S and G represent start and goal attitudes of manipulators, respectively, and line segments and circles represent arms and their joints.

The obstacles are depicted in black while the free space (F) is in white. Cases corresponding to Figures 4.1 - 4.3 are ones that heuristics based algorithms cannot generate paths efficiently. In Figures 4.1 and 4.2, two obstacles constitute narrow corridors. Figure 4.3 is an environment in which seven obstacles constitute six narrow openings. For all cases corresponding to these Figures the number of arms are changed from 7 to *18*, length of individual arms was set to *10* cm, and the free space contains four different width of narrow passages through which the manipulator must pass.

Figure 4.4 and 4.5 cases are to evaluate BFA performance in difficult but practical situations. By using obstacles placement in Figure 4.4, BFA performance under locus and attitude constraints was also evaluated. Figure 4.5 is an environment where many small obstacles are scattered. The number of obstacles and number of arms were changed to evaluate the influence of copy points and the copy propagation on the performance of BFA. BFA efficiently found collision free paths in all of these cases.

4.2.1 First Test Cases

In the first test cases, two obstacles constitute narrow corridors as shown in Figure 4.1, and the width of the corridor was changed from *15*cm to *30*cm. The number of arms are changed from 7 to *18* and lengths of all arms were set to *10*cm. Figure 4.1(a) shows the start and goal attitude of the manipulator, (b) shows the sizes of a typical environment, and (c) shows a path of all the arms obtained by BFA.

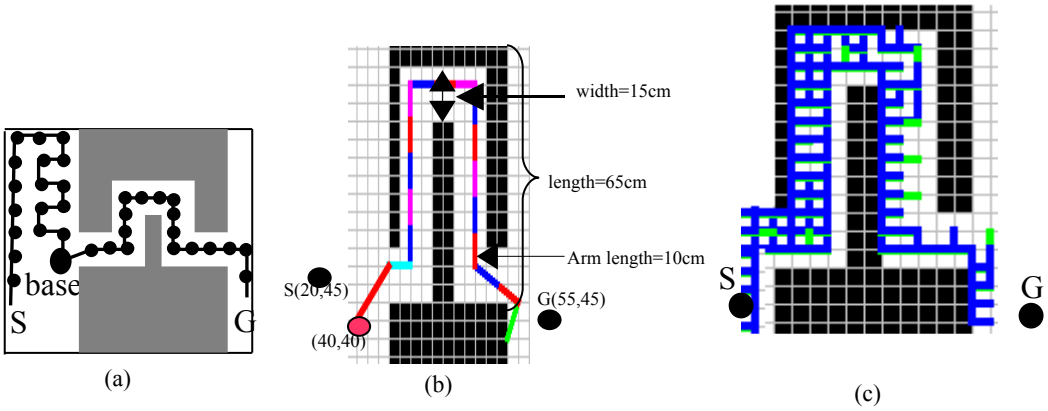


Figure 4.1 An Environment with a Narrow Corridor

4.2.2 Second Test Cases

In the second test cases, two obstacles constitute long narrow corridors with seven corners as shown in Figure 4.2 and the width of the corridor was changed from *15*cm to *30*cm. The number of arms are changed from *12* to *18* and lengths of all arms were set to *10*cm.

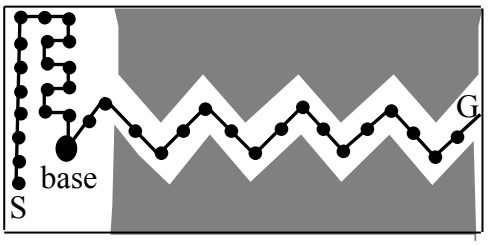


Figure 4.2 An Environment with a Long Narrow Corridor

4.2.3 Third Test Cases

In the third test case, seven obstacles constitute six narrow openings as shown in Figure 4.3 and the width of the openings was changed from *15*cm to *30*cm. The number of arms was changed from 7 to 18 and lengths of all arms were set to *10*cm. Figure 4.3 (a) shows the start and goal attitude of the manipulator, (b) shows the sizes of a typical environment, and (c) shows a path of all the arms obtained by BFA.

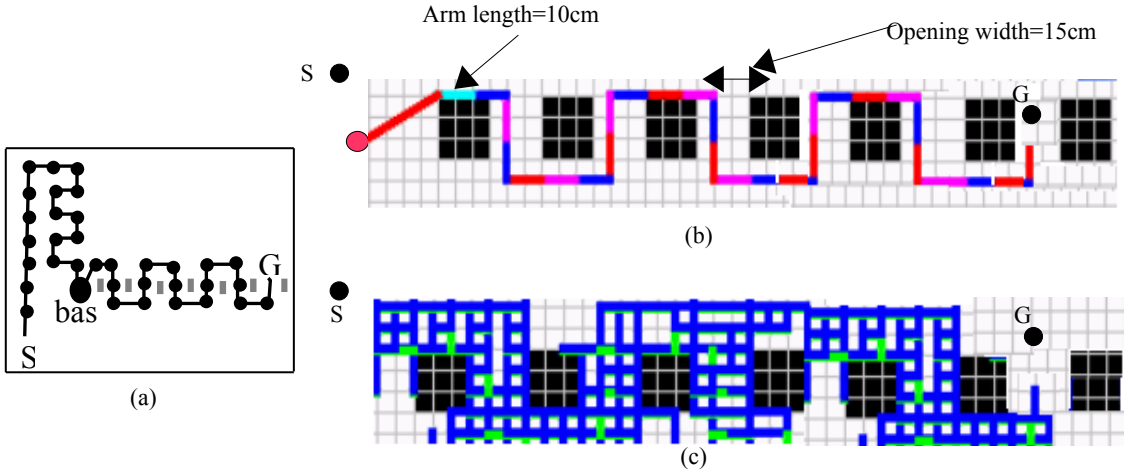


Figure 4.3 An Environment with 6 Openings

4.2.4 Forth Test Cases

Figure 4.4 cases are to evaluate BFA performance in difficult but practical situations. In theses cases, the number of arms were changed from *2* to *6* while setting the length of the *1*st and the *2*nd arms to *50* cm and *70* cm, respectively. These cases can be considered as one of the most difficult ones because the free space for the *2*nd arm is divided into *2* regions that are connected by a single point P1. In the Figure, *4* obstacles A, B, C and D are located, and the gap between A and C is set just as the same size as the length of the *2*nd arm. Therefore, the movable end of the *1*st arm must be located at single point P1 in order to change the direction of the *2*nd arm. In other words, *2* areas that include start and goal attitudes S and G are connected by just a single point P1, and finding collision free paths that connect attitudes S and G is very difficult for heuristics based algorithms. Two arcs represent the locus of the movable end of the *1*st arm; firstly it rotates from the initial position to P1 (solid arc), then moves back to P2 (dashed arc), because the manipulator cannot rotate the 2nd arm, which is initially directed to the left hand side of the *1*st arm, to the right hand side except at point P1.

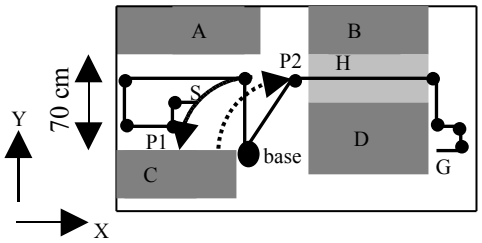


Figure 4.4 An Environment with Free Spaces Connected by a Single Point

The ability of BFA to constrain arms to follow predefined loci or to take predefined attitudes is also evaluated by using environment shown in Figure 4.4. As a locus constraint, the movable end of the last arm was enforced to follow the edge of obstacle B when its joint was inside of area H, and as an attitude constraint, the last arm was enforced to be parallel to the X-direction when its joint was in H. There are many potential applications of the constrained path generation, e.g. an end effector should keep itself vertically up all the time in order to transport a glass of water. In other applications, the last arm should be move in a plane of workspace. While several works have considered specific forms of constraints, the problem with general arm constraints has not been addressed in previous works.

4.2.5 Fifth Test Cases

Figure 4.5 cases are an environment where many small obstacles are scattered. This Figure is also to evaluate BFA performance in difficult but practical situations. In these cases, the length of all arms are set to *10*cm. The number of obstacles and number of arms were changed to evaluate the influence of copy points and the copy propagation on the performance of BFA.

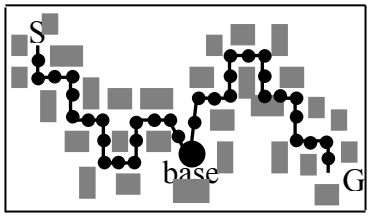


Figure 4.5 An Environment with Scattered Obstacles

4.3 Evaluation Results

Tables 4.1 and 4.2 show the evaluation results. The first and second columns of these tables represent the test environment numbers and numbers of arms, respectively. The third and fourth columns of table 4.1 indicate the number of corners or openings and the width of corridors or openings in Figures 4.1, 4.2 and 4.3. The third column of table 4.2 indicates the number of obstacles in Figures 4.4 and 4.5. The rest of columns of Tables 4.1 and 4.2 represent the total computation time, off-line part computation time, real-time part computation time, total path length, total number of copy points, the ratio of copy points to the total number of points, and the maximum number of

copy points.

Table 4.1 Evaluation Results for Cases Corresponding to Figures 4.1 - 4.3

Figure No.	Number of arms	Number of corners or openings	Corridor or opening width(cm)	Total computation time(sec)	Off-line part computation time(sec)	Real-time part computation time(sec)	Total path length	Total number of copy points	Ratio of copy points to the total number of points	Maximum number of copy points
4.1	18	2	15	18.15	2.83	15.32	25780	858	0.72	1
			20	5.68	2.83	2.85	3480	282	0.24	1
			25	5.25	2.83	2.42	2862	234	0.2	1
			30	5.17	2.83	2.34	2852	262	0.23	1
	12	2	15	4.56	2.61	1.95	2932	438	0.57	1
			20	3.44	2.61	0.83	1280	162	0.21	1
			25	3.38	2.61	0.77	890	196	0.18	1
			30	3.38	.2.61	0.77	874	142	0.17	1
	8	2	15	3.07	2.55	0.52	622	186	0.36	1
			20	2.8	2.55	0.25	434	82	0.16	1
			25	2.8	2.55	0.25	316	74	0.14	1
			30	2.8	2.55	0.25	312	66	0.12	1
4.2	18	7	15	7.16	2.97	4.18	5928	890	0.77	1
		7	20	5.15	2.97	2.18	2588	0	0	0
		7	25	4.97	2.97	1.99	2358	0	0	0
		7	30	4.26	2.97	1.29	1354	0	0	0
	12	4	15	3.6	2.64	0.96	1062	412	0.53	1
		4	20	3.36	2.64	0.72	770	0	0	0
		4	25	3.36	2.64	0.72	790	0	0	0
		4	30	3.19	2.64	0.55	622	0	0	0
4.3	18	6	15	8.1	3.16	4.94	7264	550	0.48	1
		6	20	4.72	3.12	1.6	1948	0	0	0
		6	25	4.52	3.12	1.4	1646	0	0	0
		6	30	4.36	3.12	1.24	1380	0	0	0
	12	4	15	3.62	2.7	0.92	1188	250	0.32	1
		4	20	3.25	2.69	0.56	608	0	0	0
		4	25	3.25	2.69	0.56	588	0	0	0
		4	30	3.21	2.69	0.52	528	0	0	0
	7	2	15	2.72	2.56	0.16	218	60	0.13	1
		2	20	2.71	2.55	0.16	186	0	0	0
		2	25	2.71	2.55	0.16	186	0	0	0
		2	30	2.7	2.55	0.14	124	0	0	0

Table 4.2 Evaluation Results for Cases Corresponding to Figures 4.4 - 4.5

Figure No.	Number of arms	Number of obstacles	Total computation time(sec)	Off-line part computation time(sec)	Real-time part computation time(sec)	Total path length	Total number of copy points	Ratio of copy points to the total number of points	Maximum number of copy points
4.4 (No constraints)	6	4	2.27	1.62	0.65	986	1086	2.8	10
	5	4	2.04	1.59	0.45	760	1086	3.39	10
	4	4	1.84	1.57	0.27	534	1086	4.24	10
	3	4	1.65	1.43	0.19	332	656	3.41	7
	2	4	1.46	1.33	0.12	170	129	1	3
	6	3	2.07	1.64	0.43	644	986	2.56	9
	5	3	1.91	1.61	0.3	516	986	3.08	9
	4	3	1.77	1.59	0.18	388	556	3.85	9
	3	3	1.61	1.45	0.14	268	80	2.89	6
	2	3	1.44	1.33	0.12	148	80	0.63	2
4.4 (With locus constraints)	6	4	2.25	1.62	0.63	984	1142	2.97	10
	5	4	2.02	1.59	0.43	758	1142	3.56	10
	4	4	1.83	1.57	0.26	533	1135	4.43	10
	3	4	1.64	1.43	0.18	331	660	3.43	6
	2	4	1.45	1.33	0.12	169	128	1	3
	6	3	2.06	1.64	0.42	643	1042	2.71	9
	5	3	1.89	1.61	0.2	515	1042	3.25	9
	4	3	1.78	1.59	0.18	387	1042	4.07	9
	3	3	1.61	1.45	0.14	266	582	3.03	6
	2	3	1.43	1.33	0.11	145	79	0.62	2
4.4 (With attitude constraints)	6	4	2.25	1.62	0.63	984	1162	3.02	10
	5	4	2.02	1.59	0.43	758	1162	3.63	10
	4	4	1.83	1.57	0.26	533	1162	4.68	10
	3	4	1.64	1.43	0.18	331	675	3.51	6
	2	4	1.45	1.33	0.12	169	128	1	3
	6	3	2.06	1.64	0.42	643	1042	2.71	9
	5	3	1.89	1.61	0.2	515	1042	3.25	9
	4	3	1.78	1.59	0.18	387	1042	4.07	9
	3	3	1.61	1.45	0.14	266	582	3.03	6
	2	3	1.43	1.33	0.11	145	79	0.62	2
4.5	18	29	7.87	3.58	4.28	6210	4440	3.85	17
		24	7.39	3.58	3.8	5214	3387	2.94	17
		19	6.51	3.58	2.93	4030	2072	1.79	17
		14	5.72	3.58	2.14	2910	724	0.63	2
	12	29	4.02	3.03	0.99	1336	1804	2.34	10
		24	4.01	3.01	1	1170	1305	1.69	9
		19	3.81	3.03	0.78	930	842	1.09	8
		14	3.66	3.03	0.63	688	364	0.47	2
	6	29	2.76	2.63	0.13	196	209	0.82	5
		24	2.75	2.63	0.12	195	83	0.21	4
		19	2.75	2.63	0.12	152	67	0.17	3
		14	2.73	2.63	0.1	132	28	0.07	2

4.3.1 Performance Comparisons between PRM

The performance of BFA was compared with that of an existing algorithm. Probabilistic roadmap planner (PRM) [82] was selected as the algorithm to be compared with, because PRM is considered as one of the most efficient algorithms. Figures 4.1 and 4.3 show the obstacle placements used in the

comparisons. Computation volume and the relation between computation volume and the workspace visibility are compared between BFA and PRM. The performance of PRM depends on visibility of the workspace, i.e. PRM can find paths quickly when many point pairs are visible, however it cannot work efficiently when they are not visible. Visibility in cases corresponding to Figures 4.1- 4.3 is not high enough for heuristics based algorithms including PRM, and they cannot generate paths efficiently. BFA efficiently found collision free paths in all cases.

Tables 4.3-4.4 and Figures 4.6 show advantages of BFA when compared with PRM [79, 80]. Table 4.3 is the comparison of the computation time of BFA and PRM for cases corresponding to Figures 4.1 and 4.3. PRM requires the computation time *98.2*sec. and *231.1*sec. for cases where the manipulator has *8* and 7 arms. BFA requires only *3.07*sec. and *2.72*sec. for these cases. Although lengths of arms and width of the corridor are not specified in [79], it is apparent that BFA can calculate paths in much shorter time than PRM.

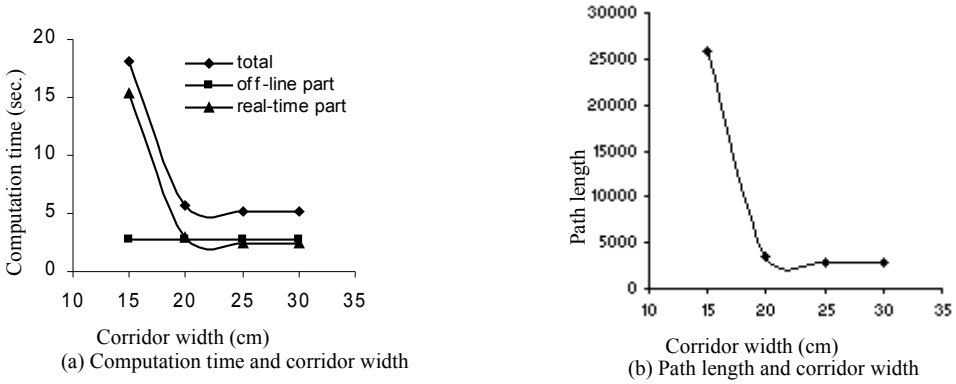
Table 4.3 Performance Comparison between PRM and BFA (for Figures 4.1 and 4.3 cases)

			PRM by Java on a 2.8GHz CPU	BFA by Java on a 1.53GHz CPU		
Figure	Number of arms	Number of openings	Total Computation time(sec)	Length of arm(cm)	Width of corridor(cm)	Total Computation time(sec)
4.1	8		98.2	10	15	3.07
4.3	7	2	231.1	10	15	2.72

Table 4.4 is the comparison results for Figure 4.2 cases. The performance of PRM is the one for a small disc manipulator, and width of the corridor in PRM test cases is calculated based on the equation shown in [80] while assuming the radius of the disc manipulator is *10*cm. While the computation time of PRM increases rapidly with the decrease of the corridor width, that of BFA does not change when the width of the corridor is more than *20*cm. Figure 4.6 (a) shows that different from PRM the total and real-time part computation times do not change with the width of the corridor when it is more than *20*cm. Although they increase when the corridor width is *15*cm, this is because of the path length as shown in Figure 4.6 (b). The *15*cm width is almost the smallest gap for 10cm arms to behave within it. Therefore they should change their attitudes frequently (as discussed later, any path planning algorithm has parts, of which computation volume increases with path length). The off-line part computation time is constant always even for these cases.

Table 4.4 Performance Comparison between PRM and BFA (for Figure 4.2 cases)

	PRM by C++ on a 1GHz CPU with 1GB RAM			BFA by Java on a 1.53GHz CPU with 224MB RAM		
Length of arm(cm)	Number of arms	Width of corri-dor(cm)	Total Com-putation time(sec)	Number of arms	Width of corri-dor(cm)	Total Com-putation time(sec)
10				18	15	7.16
10	2	20	1659	18	20	5.15
10	2	20.16	346			
10	2	20.24	160	18	25	4.96
10	2	28.5	5.23	18	30	4.26
10	2	40	1.32	18	40	4.16
10	2	67	0.87	18	60	4.16



(a) Computation time and corridor width

(b) Path length and corridor width

Figure 4.6 BFA Computation Volume and Corridor Width

When path does not exist in Figure 4.2 cases, BFA requires only the off-line part computation time. On the other hand, heuristics based algorithm cannot determine the path existence exactly; it requires *108000*sec. [80] to abandon computations.

4.3.2 BFA Performance in Various Environments

Evaluation results of cases corresponding to Figure 4.4 show that BFA performs stably also in difficult but practical situations. BFA found paths successfully and efficiently even in these cases. Namely as shown in Table 4.2, BFA found paths within *2.5* sec. for *2* to *6* arms manipulators. As shown in Figure 4.4, the manipulator had firstly moved the movable end of the *1*st arm from P2 to P1, and then moved it back to P2, in order to change its attitude from S to G.

Figure 4.5, where many small obstacles are scattered, is also an environment that is difficult to find paths for heuristics based algorithms. BFA found paths efficiently also in these cases, i.e. as shown in table 4.2, less than *8*sec. is enough even for *18* arms manipulator to find paths that avoid collision with 29 obstacles.

4.3.3 BFA Performance Comparison for Attitudes Constrained Path Generation

The advantage of BFA is that both locus and attitude constraints can be incorporated in a straightforward and intuitive way, i.e. they can be incorporated only by deleting attitudes that do not satisfy the constraints from feasible attitude sets. In Figure 4.7, A(F, *N*), a feasible attitude set of the *N*-th (last) arm at point F is an arc (U, W) when there is no constraint, and the locus constraint can be incorporated by only deleting points that do not satisfy the constraint. Because the constraint is that the movable end of the *N*-th arm should follow the edge of obstacle B, it is enough only to delete points that are not on the edge of obstacle B from A(F, *N*). Then, A(F, *N*) is reduced to a single point U, and as a consequence, paths automatically follow the edge when the joint of the *N*-th arm moves within H, because BFA generates paths by connecting only points included in FASs.
An attitude constraint can be incorporated in the same way. In the the case where the *N*-th arm should be parallel to the X-direction, A(F, *N*) is reduced to a single point V, and the *N*-th arm attitude on the path automatically becomes parallel to the X-direction when the joint of the *N*-th arm moves within H. As shown above, attitude constraints are easier to incorporate than locus constraints. Points on FASs to be deleted can be determined without considering path positions to be followed as in locus constraints.
ATACE (Alternate Task-space And C-space Exploration) [84] generates paths under general end-effecter constraints by searching Task-space (T-space) and C-space alternately. Here, T-space is the *3*-dimensional Euclidean space where the end-effecter works. It explores T-space for feasible end-effecter paths under given constraints while transforming these constraints into end-effecter velocity constraints, and then tracks these end-effecter paths in C-space in order to generate paths for the whole manipulator that are consistent to ones for the end-effecter. During the above path generation processes, ATACE constructs a search tree. Therefore in contrast to BFA, frequent calculations for converting T-space to C-space and for transforming attitude constraints to velocity constraints are required, and off course backtracks may occur frequently. Also ATACE is applicable only to end-effecter constraints. Apparently in BFA, loci and attitudes of general arms can be constrained in the same way as those of the last arm. As shown in Table 4.2, the off-line part computation time does not

change with constraints. The real-time part computation time decreases when paths and attitudes are constrained, because path lengths decrease when paths are constrained.

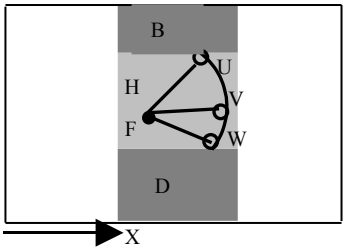
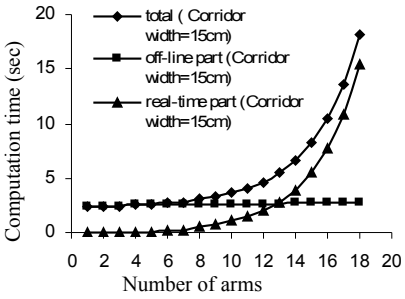


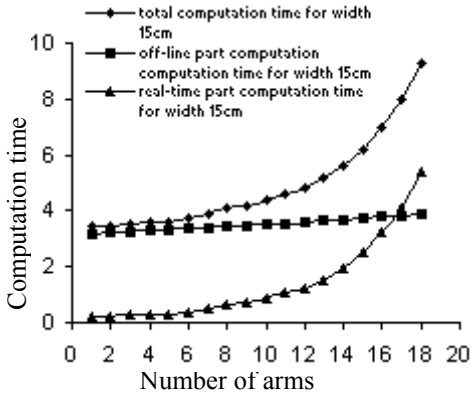
Figure 4.7 Locus and Attitude Constraints of an Arm

4.3.4 Advantage of BFA

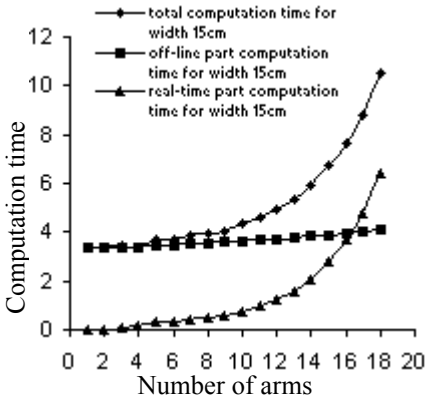
This sub-section discusses advantages of BFA. Figures 4.8 shows the most important advantages of BFA. Namely, the off-line part computation time is proportional to the number of arms. In spite of the fact that BFA is resolution complete, the off-line part computation time does not increase exponentially. This means that the number of copy points can be maintained small enough as discussed later in this sub-section. Regarding to the real time part computation time and the total computation time, they are not proportional to the number of arms. However, this is because that path length increases not linearly with the number of arms as shown in Figure 4.10. Here, it is obvious that any algorithm has parts that require the computation volume at least proportional to the path length; and according to Figure 4.9 the real-time part computation time of BFA increases just linearly with the path length. Consequently, the total computation time of BFA can be suppressed at linear order of the number of arms provided that path length increases also linearly with the number of arms. The above fact leads another advantage of BFA, i.e. it can be divided into the off-line and the real-time parts, of which computation times are dependent only on the number of arms, and only on the path length, respectively. Therefore, it becomes possible to make computation volume necessary when obstacle placement is changed independent of path length, and that necessary when goal attitudes are given independent of the number of arms.



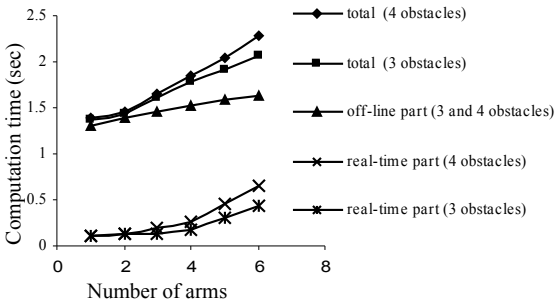
(a) Cases in Figure 4.1



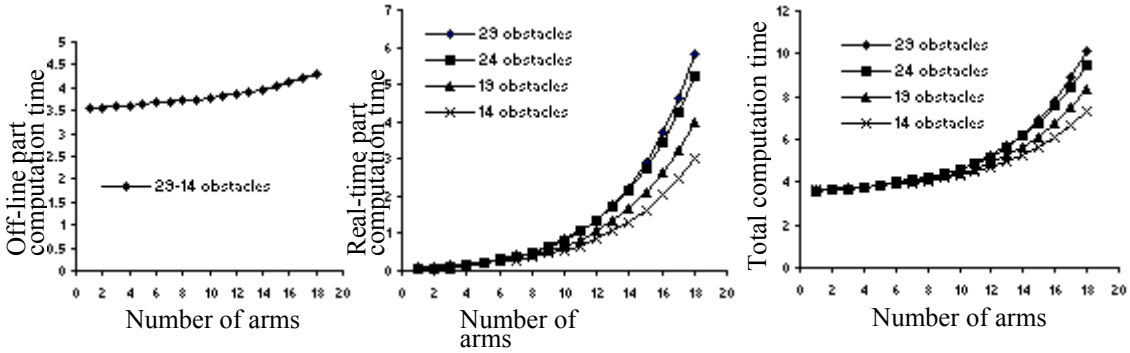
(b) Cases in Figure 4.2



(c) Cases in Figure 4.3



(d) Cases in Figure 4.4

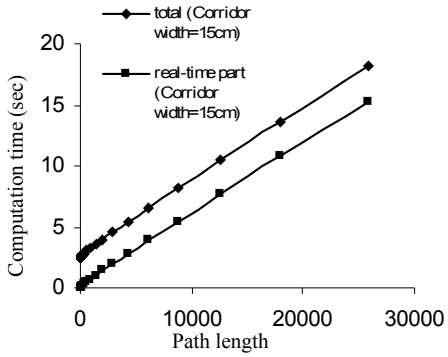


(e) Cases in Figure 4.5

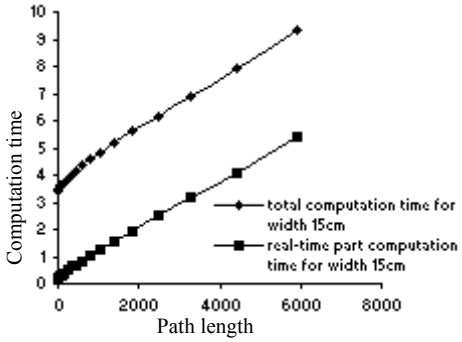
Figure 4.8 Computation Time and Number of Arms

Figures 4.8 (d) and (e) show the other advantage of BFA, i.e. BFA performance is not sensitive to environments. When obstacle A is removed from Figure 4.4, the gap that divides the free space of the *2*nd arm disappears. Therefore for heuristics based algorithms, computation times necessary for these *2* environments, the one where A is allocated (*4* obstacles cases in Figure 4.8 (d)) and the other where A is removed (3 obstacles cases), differ extremely. In contrast, Figure 4.8 (d) shows that BFA

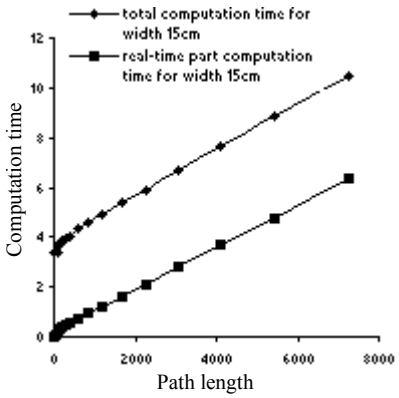
can find paths with almost the same time regardless of environments.



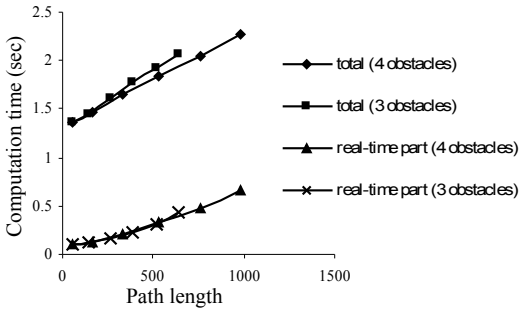
(a) Cases in Figure 4.1



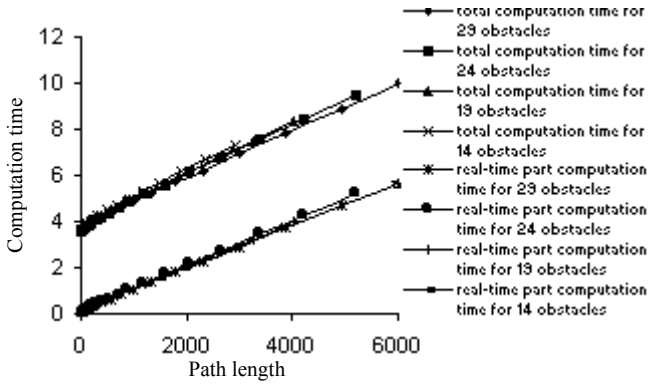
(b) Cases in Figure 4.2



(c) Cases in Figure 4.3



(d) Cases in Figure 4.4



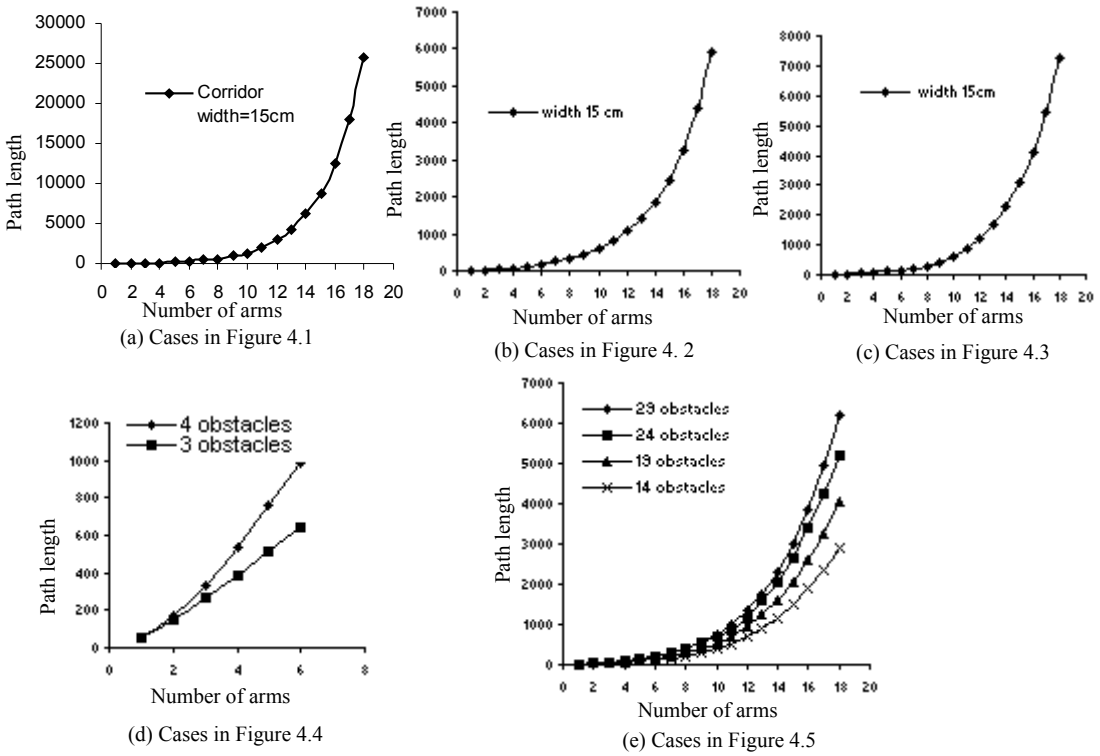
(e) Cases in Figure 4.5

Figure 4.9 Computation Time and Path Length

Table 4.2 shows that the off-line part computation times for a *6*-arms manipulator are about *1.6*sec. in both environments, the real-time part computation times are *0.65*and *0.43*sec., and the total computation times are *2.27* and *2.07*sec. for cases where obstacle A is allocated and removed. Namely, the off-line part computation time is constant even environments change, and although the

real-time part computation time changes with environments, it is because that the *1*st arm should move its movable end between P1 and P2 twice when obstacle A exists. For Figure 4.5 cases, the off-line part computation time does not change even the number of scattered obstacles is changed fron 14 to 29. As explained in the previous section, the off-line part computation time for Figure 4.1-4.3 cases does not change even when the width of the corridor changes.

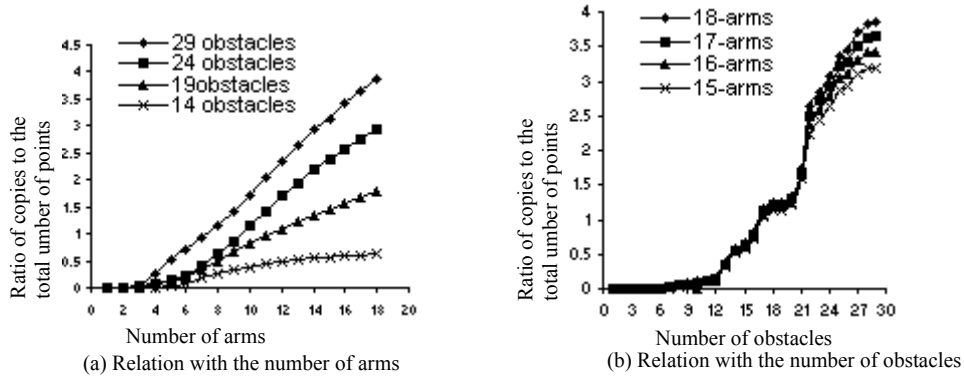
Figures 4.11 is the evaluation results of cases corresponding to Figure 4.5, and shows that copy of points generated in BFA do not cause serious problems even in complicated cases. As shown in Figures 4.11 (a), (b) and Table 4.2, the ratio of copy points increases less than linearly with the number of arms and the number of obstacles, despite the fact that copy points generated in the higher arm spaces propagate to the lower arm spaces. The difference of the ratio of copy points to the total number of points are *1.51* and *1.52*% between a *18* and a *12* arms manipulators, and between a *12* and a *6* arms manipulators, respectively for *29* obstacles cases. Regarding to the number of obstacles, the difference is *0.91*% between a *29* and a *24* obstacles cases, and *1.15*% and *1.16%* between a *24* and a *19*, and a *19* and a *14* obstacles cases, when the number of arms is *18*.



(a) Cases in Figure 4.1 (b) Cases in Figure 4. 2 (c) Cases in Figure 4.3

(d) Cases in Figure 4.4 (e) Cases in Figure 4.5

Figure 4.10 Path Length and Number of Arms

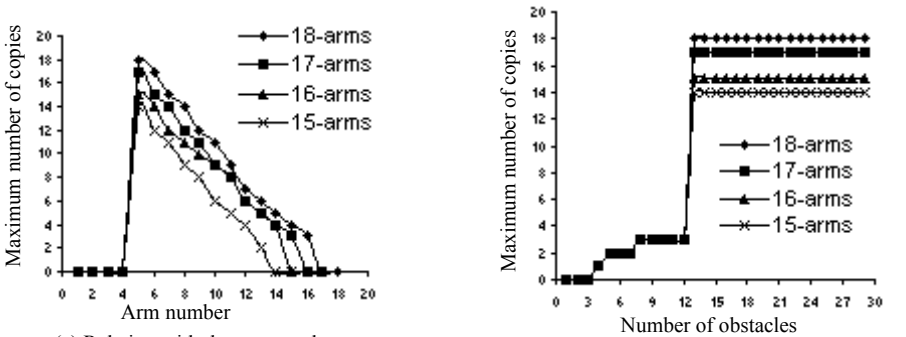


(a) Relation with the number of arms

(b) Relation with the number of obstacles

Figure 4.11 Copy Points in Figure 4.5 Cases

Although, the maximum number of copy points increases when the arm number becomes small as shown in Figure 4.12 (a), because of the copy propagation, it saturates with the number of obstacles as shown in Figure 4.12 (b). Therefore, the ratio of copy points to the total number of points are small enough, i.e. it is suppressed at *3.85%* for a *18* arms manipulator even when *29* obstacles are scattered, and computation volume does not increase so much with the number of obstacles as shown in Figure 4.13. Figure 4.13 and Table 4.2 show that the off-line part computation time is almost the same even the number of obstacles increases. Although the real-time part computation time increases with the number of obstacles, it is only because paths have long lengths when they avoid many obstacles. As a conclusion, copies generated in BFA do not reduce its performance seriously. BFA can maintain its performance at the theoretical level even many obstacles are scattered.



(a) Relation with the arm number

(b) Relation with the number of obstacles

Figure 4.12 Maximum Number of Copy Points in Figure 4.5 Cases

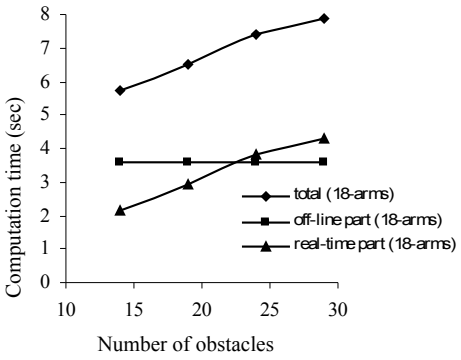


Figure 4.13 Computation Time and Number of Obstacles in Figure 4.5 Cases

4.4. Conclusion

The performance of BFA has been evaluated for manipulators with many arms that were operated in various *2*-dimensional environments. The most important advantage of BFA is that its computation volume is proportional to the number of arms; therefore it can be applied to manipulators with many arms. Moreover BFA performance is not sensitive to the environments, i.e. the off-line part computation time is almost constant even environments change. Although the real-time part computation time of BFA increases when environments become complicated, this is because that path length increases.

Evaluation results show that the ratio of copy points generated in BFA to the total number of points is less than *4*%. This means that the number of copy points does not decrease the performance of BFA seriously. Therefore BFA can maintain its performance at the level theoretically expected. As the consequence, BFA calculates paths with much shorter time than existing heuristics based algorithm in complicated cases. BFA also enables easy locus and attitude constrained path calculations.

Chapter 5

Conclusion

The performance of BFA has been evaluated for manipulators with many arms that were operated in various *2*-dimensional environments. BFA is a resolution complete and backtrack free algorithm, and its computation time and memory space are the order of *NMR*. Here, *M* and *N* are the number of grid points that cover the workspace of the manipulator, and the number of arms, respectively. *R* is the upper bound of the number of grid points on the moving ranges of individual arms.

This is the most important advantage of BFA, i.e. its computation volume is proportional to the number of arms; therefore it can be applied to manipulators with many arms. Moreover BFA performance is not sensitive to the environments, i.e. the off-line part computation time is almost constant even environments change. Although the real-time part computation time increases when environments become complicated, this is because that path length increases.

Evaluation results show that the ratio of copy points generated in BFA to the total number of points is less than *4*%. This means that the number of copy points does not decrease the performance of BFA seriously. Therefore BFA can maintain its performance at the level theoretically expected. BFA also enables easy locus and attitude constrained path calculations.

In conclusion, advantages of BFA can be summarized 1) its computation volume is proportional to the number of arms despite that BFA is resolution complete. 2) its computation time is not sensitive to environments, 3) it is easy to generate locus or attitude constrained paths, and 4) it is a resolution complete and backtrack free algorithm. As the consequence, BFA calculates paths with much shorter time than existing heuristics based algorithm in complicated cases.

The followings are future works to make BFA more practical.

- Improvement of the real-time part performance.
  The current BFA generates redundant paths. Although it is not difficult to remove redundant parts from given paths, they must be removed in the path generation process, because the performance of the real-time part is dependent on path lengths. To reduce redundant paths, strategies to select connecting point pairs of neighboring FASs among of multiple

possibilities during path generation processes must be established.

- Program development for *3*-D workspace.

  Many manipulators work in *3*-D environments, therefore it is inevitable to implement BFA for *3*-D applications. Because the volume of data increases drastically in *3*-D applications compared with *2*-D applications, sophisticated data structure must be developed.

- Development of a multi manipulator collaboration algorithm.

  Complicated manufacturing processes can be accomplished only through the cooperation among multiple manipulators. In order to convey a long work piece stably, *2* or *3* manipulators are necessary for example. To make manipulators applicable to various and important applications, efficient path planning algorithms for multiple manipulators become necessary. In this regard, BFA has a substantial advantage, i.e. in BFA [86,] path planning for multi manipulators can be executed almost completely in parallel.

References

[1] T. Lozano-Perez and M.A. Wesey, “An Alogorithm for Planning Collision-free Paths among Polyhedral Obstacles,” CACM, Vol. 22, No. 10, pp. 560-570, 1979.

[2] J. H. Reif, “Complexity of the mover’s problem and generalizations,” In Proceedings IEEE Symposium of Foundations of Computer Sci-ence, pp. 421–427, San Juan, Puerto Rico, Oct. 1979.

[3] D. G. Kirkpatrick, “Efficient computation of continuous skeletons,” In the 20th Symposium on the Foundations of Computer Science, pp. 18-27, 1979.

[4] T. Asano, Guibas, L., Hershberger, J., and Imai, H, “Visibility-polygon search and Euclidean shortest path,” In The 26th Symposium of Foundations of Computer Science, Portland, pp. 155-164, Oct. 1985.

[5] C. H. Papadimitriou, “An algorithm for shortest-path motion in three dimensions,” Int. Process Lett., pp. 259-263, 1985.

[6] O. Khatib, “Real-time Obstacle Avoidance for Manipulator and Mobile Robots,” in Proceedings of IEEE Int. Conf. on Robotics and Automation, pp. 500-505, 1985.

[7] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” The Int. Journal of Robotics Research, Vol. 5(1), pp. 90–98, 1986.

[8] B. Faverjon and P. Tournassoud, “A Local Based Approach for path Planning of Manipulators with a high Number of Degrees of Freedom,” IEEE Conf. On Robotics and Automation, pp. 1152-1159, 1987.

[9] W. Gordon, “Motion planning in the Presence of movable obstacles,” In ACM Symp. Computat. Geometry, pp 279-288, ACM Press, New York, NY, 1988.

[10] J. F. Canny, “The complexity of the robot motion Planning,” MIT Press: Cambridge, MA,

1988.

[11] P. Khosla and R. Volpe, “Super quadric artificial potentials for obstacle avoidance and approach,” In IEEE Int. Conf. on Robotics and Automation, pp. 1778–1784, 1988.

[12] J. Barraquand and J.C. Latombe, “A Monte-Carlo algorithm for path planning with many degrees of freedom,” in Proceedings of IEEE Int. Conf. on Robotics and Automation, pp. 1712-1717, Cincinnati, OH, 1990.

[13] K. K. Gupta, “Fast Collision Avoidance for Manipulator Arms: A Sequential Search Strategy,” IEEE Trans. on Robotics and Automation, Vol. 6, pp. 522-532, 1990.

[14] J. Lengyel, M. Reichert, B.R. Donald and D. P. Greenberg “Real-time robot motion planning using rasterizing computer graphics hardware,” Computer Graphics, Vol. 24(4), pp. 327–335, 1990.

[15] C. I. Connolly and J. B. Burns, “Path planning using Laplace's equation,” In IEEE Int. Conf. on Robotics and Automation, pp. 2102–2106, 1990.

[16] J. Barraquand and J. C. Latombe, “Robot Motion Planning: A Distributed Representation Approach,” IJRR, pp. 628-649, 1991.

[17] J.C. Latombe, “Robot Motion Planning,” New York: Kluwer, 1991.

[18] F. Aurenhammer, “Voronoi diagrams-A survey of fundamental geometric data structure,” ACM Comput. Surv. Vol. 23, pp. 345-405, Sept. 1991.

[19] E. Rimon, “A navigation function for a simple rigid body,” In IEEE Int. Conf. on Robotics and Automation, pp. 546–551, 1991.

[20] Y. Koga and J. C. Latombe, “Experiments in Dual-Arm Manipulator Planning,” Proc. Of the IEEE Int. Conf. On Robotics and Automation, pp. 2238-2245, 1992.

[21] L. Y. Hwang and A. Narendra, “Gross motion planning-A Seurvey,” ACM Comput. Surv., Vol. 24, No.3, pp. 219-292, Sept. 1992.

[22] P. C. Chen, and Y. K. Hwang, “SANDROS: A motion planner with performance proportional to task difficulty,” In Proceedings of the IEEE Int. Conf. on Robotics and Automation, pp. 2346-2353, 1992.

[23] J. Kim and P. Khosla, “Real-time obstacle avoidance using harmonic potential functions,” IEEE Transactions on Robotics and Automation, Vol. 8(3), pp. 338–349, 1992.

[24] E. Rimon and D. E. Koditschek, “Exact robot navigation using artificial potential functions,” IEEE Trans. on Robotics and Automation, Vol. 8(5), pp. 501–17, 1992.

[25] J. Barraquand, B. Langlois and J. Latombe, “Numerical potential field techniques for robot path planning,” IEEE Trans. on Systems, Man Cybernetics, Vol. 22(2), pp. 224–241, 1992.

[26] A. Graham and R. Buckingham, “Real time collision avoidance of manipulators with multiple redundancy,” Mechatronics, Vol. 3(1), pp. 89–106, 1993.

[27] F. Janabi-Sharifi and D. Vinke, “Robot path planning by integration the artificial potential field approach with simulated annealing,” In IEEE Int. Conf. on Robotics and Automation, pp. 282–287, 1993.

[28] L. E. Kavraki and J. C. Latombe, “Randomized preprocessing of configuration space for fast path planning,” In IEEE Int. Conf. on Robotics and Automation, Vol. 3, pp. 2138–2145, San Diego, CA, USA, May 1994.

[29] T. Laliberté, “Planification de trajectoire d’un manipulateur sériel redondantdans un environement encombré,” Master Degree Thesis, Faculté des Sciences et de Génie, Université Laval, QC, Canada, 1994.

[30] A. Denker, D. Atherton, “No-overshoot control of robotic manipulators in the presence of obstacles,” Journal of Robotic Systems, Vol. 11(7), pp. 665–78, 1994.

[31] L. Kavraki, “Random networks in configuration space for fast path planning,” PhD thesis; 1994, Stanford University.

[32] A. Hayashi, “Geometric motion planning for highly redundant manipulators using a continuous model,” PhD thesis, The University of Texas at Austin, 1994.

[33] K. K. Gupta and Z. Guo, “Motion Planning for Many Degrees of Freedom: Sequential Search with Backtracking,” IEEE Trans. on Robotics and Automation, Vol. 11, pp. 897-906, 1995.

[34] P. Svestka and M. H. Overmars, “Coordinated motion planning for multiple car-like robots using probabilistic roadmaps,” In IEEE Int. Conf. on Robotics and Automation, Vol. 2, pp. 16311636, Nagoya, Japan, May 1995.

[35] T. Nishimura, K. Sugawara, I. Yoshihara and K. Abe, “A Motion Planning Method for a Hyper Multi-joint Manipulator using Genetic Algorithm,” in Proceedings IEEE Int. Conf. on Systems, Man and Cybernetics, Vol. 4, pp. 645-650, Tokyo, Japan, Oct. 1995.

[36] Y. Kitamura, T. Tanaka, F. Kishino and M. Yachida, "3-D planning in a dynamic environment using an octree and an artificial potential field," in Proceedings of IEEE Int. Conf. on Intelligent Robots and Systems , vol. 2, pp. 474-479, 1995.

[37] K. H. Woong and S. Hong, “A Hierarchical Collision-Free Path Planning Algorithm for Robotics,” Proceedings of the Int. Conf. on Intelligent Robots and Systems (IROS), pp. 488-495, 1995.

[38] K. Noriyuki and K. Taguchi, “Collision-Free Path Planning for a Manipulator Using Free Form Surface,” Proceeding of the Int. Conf. on intelligent robots and system(IROS), pp. 130- 137, 1995.

[39] M. Tarokh, **“**Fast Path Planning for Robot Manipulators by Formation-Posture Decomposition,” Proceedings of the Int. Conf. on Intelligent Robots and Systems (IROS), pp138-143, 1995.

[40] K. K. Gupta and X. Zhu, “Practical global motion planning for many degrees of freedom: a novel approach within sequential framework,” Journal of Robotics Systems, Vol. 12(2), pp. 105–109, 1995.

[41] H. Chang and T. Y. Li”Assembly maintainability study with motion planning,” in Proc. IEEE Int. Conf. Robot. Autom. pp. 1012-1019, 1995.

[42] L. E. Kavraki, P. Svestka, J. C. Latombe and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” IEEE Trans. on Robotics and Automation, Vol. 12(4), pp. 566–580, Aug. 1996.

[43] H. Chang, “A New Technique To Handle Local Minimum For Imperfect Potential Field Based Motion Planning,” in Proceedings of IEEE Int. Conf. on Robotics and Automation, pp. 108-112, Minneapolis, MN, April 1996.

[44] H. Chang, “A new technique to handle local minimum for imperfect potential field based motion planning”, In IEEE Int. Conf. on Robotics and Automation, pp. 108–112, 1996.

[45] A. Mclean and S. Cameron, “The virtual springs method: path planning and collision avoidance for redundant manipulators,” The Int. Journal of Robotic Research, Vol. 15(4), pp. 300–319, 1996.

[46] E. S. Conkur and R. Buckingham, “Increasing the maneuvering ability of highly redundant manipulators,” in Proceedings of IEEE Int. Conf. on Robotics and Automation, Albuquerque, NM, pp. 155-160, 1997.

[47] V. Moreno, E. Sanz and F. J. Blanco, “Parallel Path Planning with Temporal Parameterization,” Proceeding of IEEE Int. Symposium on Computational Intelligence in Robotics and Automation(CIRA) , pp 102-107, 1997.

[48] J. M . Ahuactzin and K. K Gupta, “A motion planning based approach for inverse kinematics of redundant robots: the kinematic roadmap,” In IEEE Int. Conf. on Robotics and Automation, pp. 3609–3614, 1997.

[49] E. S. Conkur and R. Buckingham, “Manoeuvring highly redundant manipulators,” Robotica, Vol. 15(4), pp. 435–447, 1997.

[50] K. K. Gupta, “Motion Planning for Flexible Shapes (Systems with Many Degrees of Freedom): A Survey,” The Visual Computer, Vol. 14. No.5-6, pp. 288-302, 1998.

[51] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones and D. Vallejo, “OBPRM: An Obstacle-based PRM for 3D Workspaces,” Proc. 3rd Workshop on Algorithmic Found. Robot, pp. 155-168, 1998

[52] M. Z. ZBonert, L. H. Shu and B. Benhabib, “ Motion Planning for Multi-Robot Assembly Systems,” Proc. Of the 1999 ASME Design Engineering Technical Conf., 1999.

[53] E. J. Solteiro Pires and J. A. Tenreiro Machado, “A Trajectory Planner for Manipulators Using Genetic Algorithms,” in Proceedings of IEEE Int. Symposium on Assembly and Task Planning, pp. 163-168, Portugal, 1999.

[54] M. Piaggio and A. Sgorbissa, “AI-CART: an Algorithm to Incrementally Calculate Artificial potential fields in Real-Time,” in Proceedings of IEEE Int. Symposium on Computational Intelligence in Robotics and Automation, pp. 238-243, Monterey, CA, November 1999.

[55] V. Boor, M. H. Overmars and A. F. van der Stappen, “The Gaussian Sampling Strategy for Probabilistic Roadmap Planners,” Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1018-1023, 1999.

[56] D. Hsu, J. C. Latombe and R. Motwani, “Path Planning in Expansive Configuration Spaces,” Int. J. Computat. Geom. Applic.,Vol. 9, No. 4-5, pp. 495-512, 1999.

[57] C. Nissoux, T. Siméon, and J. P. Laumond, “Visibility-based Probabilistic Roadmaps,” Proc. IEEE/RSJ Int. Conf. on Intell. Robots Syst., pp. 1316-1321, 1999.

[58] R. Bohlin and L. E. Kavraki, “Path planning using lazy PRM”, In IEEE Int. Conf. on Robotics and Automation, Vol. 1, pp. 521–528, San Francisco, CA, USA, Apr. 2000.

[59] L. Chengqing, M.H. Ang Jr, H. Krishnan and L.S. Yong, “Virtual Obstacle Concept for Local-minimum-recovery in Potential-field Based Navigation,” in Proceedings of IEEE Int. Conf. on Robotics and Automation, pp. 983-988, San Francisco, CA, April 2000.

[60] J. J. Kuffner and S. M. LaValle, “Rrt-connect: An Efficient Approach to Single-Query Path Planning,” IEEE Int. Conf. On Robotics and Automation, 2000.

[61] R. Bohlin and L. E. Kavraki, “Path Planning Using Lazy PRM,” Proc. IEEE Int. Conf. on Robotics and Automation, pp. 521-528, 2000.

[62] R. Bohlin and L. E. Kavraki, “A randomized algorithm for robot path planning based on lazy evaluation”, Handbook on randomized computing. Kluwer Academic Publishers, 2001.

[63] S. Caselli, M. Reggiani and R. Rocchi, “Heuristic Methods for Randomized Path Planning in Potential Fields,” in Proceedings of IEEE Int. Symposium on Computational Intelligence in Robotics and Automation, pp. 426-431, Banff, AB, 2001.

[64] M.G. Park, J.H. Jeon and M.C. Lee, “Obstacle Avoidance for Mobile Robots using Artificial Potential Field Approach with Simulated Annealing,” in Proceedings of Int. Symposium on Industry Electronics, Vol. 3, pp. 1530-1535, Pusan, South Korea, Jun 2001.

[65] H. Arjangand M. Tarokh, “Manipulator Path Planning by decomposition: Algorithm and Analysis,” IEEE Trans. on Robotics Automation, Vol, 17, No.6, 2001.

[66] X. Ji and J. Xiao, “Planning Motion Compliant to Complex Contact States”, Int. Journal of Robotics Research, Vol. 20, No. 6, pp. 446-465, 2001.

[67] L. K. Dale and N. M. Amato, “Probabilistic Roadmaps-Putting it all Together,” Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1940-1947, 2001.

[68] M. Foskey, M. Garber, M. C. Lin and D. Manocha, “A Voronoi-based Hybrid Motion Planner”, Proc. IEEE/RSJ Int. Conf. on Intell. Robots Syst., pp. 55-60, 2001.

[69] S. M. La Valle and J. J. Kuffner, “Randomized kinodynamic Planning,” Int. J. Robotics Research, Vol. 20, No. 5, pp. 278-400, 2001.

[70] G. Oriolo, M. Ottavi and M. Vendittelli, “Probabilistic Motion Planning for Redundant Robots along Given End-Effector Paths,” in Proceedings of Int. Conf. on Intelligent Robots and Systems, Vol. 2, pp. 1657-1662, Lausanne, Switzerland, 2002.

[71] G. Lian, Z. Sun and K. Kab II, “Smart Collision Free Motion Control for Robot Arms,” in Proceedings of 4th World Congress on Intelligent Control and Automation, pp. 2817-2821, Shanghai, China, 2002.

[72] S. Caselli, M. Reggiani and R. Sbravati, “Parallel Path Planning with Multiple Evasion Strategies,” in Proceedings of IEEE Int. Conf. on Robotics and Automation, pp. 260-266, Washington, DC, 2002.

[73] P. Leven and S. Hutchison, “Using Manipulability to Bias Sampling During the Construction of Probabilistic Roadmaps,” in Proceedings of IEEE Int. Conf. on Robotics and Automation, Washington DC, USA, pp. 2134-2140, 2002.

[74] C.C. Lin and J.-H. Chuang, “Potential-Based Path Planning for Robot Manipulators in 3-D Workspace,” in Proceedings of IEEE Int. Conf. on Robotics and Automation, vol. 3, pp. 3353-3358, Taipei, Taiwan, 2003.

[75] S. Ando, “A fast collision-free path planning method for general robot manipulator,” in Proceedings of 2003 IEEE Int. Conf. on Robotics and Automation, pp. 2871-2877, Taipei, Taiwan, 2003.

[76] S. R. Lindemann and S. M. LaValle, “Incremental Low-discrepancy Lattice Methods for Motion Planning,” Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2920-2927, 2003.

[77] S. Tamura, T. Yanase, Md. Nazrul Islam**,** T. Ito and H. Miyashita “A New Path Planning Algorithm for Manipulators,” IEEE Int. Conf. on Systems, Man and Cybernetics, Waikoloa, Hawaii, USA, pp. 2242-2247, 2005.

[78] E. S. Conkur, R. Buckingham and A. Harrison, “The Beam Analysis Algorithm for Path Planning for Redundant Manipulators,” Mechatronics, Vol. 15, pp. 67-94, 2005.

[79] Z. Sun, D. Hus, T. Jiang, H. Kurniawati and J. H. Reif, “Narrow Passage Sampling for Probabilistic Roadmap Planning,” IEEE Trans. on Robotics and Automation, Vol. 21, No. 6, pp. 1105-1115, 2005.

[80] M. Saha and J. C. Latombe, “Finding Narrow Passages with Probabilistic Roadmaps: The Small-Step Retraction Method,” Autonomous Robots, Vol. 19, pp. 301-319, 2005.

[81] Md. Nazrul Islam, T. Murata, S. Tamura, and T. Yanase “Evaluation a New Backtrack Free Path Planning Algorithm for Multi-Arm Manipulators,” Asian simulation conf. JSST, Tokyo, Japan, pp.133-137, 2006.

[82] D. Hsu, J. C. Latombe, Hanna Kurniawati, “On the probabilistic foundations of probabilistic roadmap planning,” The Int. journal of robotics research, Vol. 25, No. 7, pp. 627-643, 2006.

[83] K. D. Santosha and P. Eberhard, “Dynamic Analysis of Flexible Manipulators, a Literature Review,” Mechanism and Machine Theory, 41, pp. 749-777, 2006.

[84] Z. Yao and K. K. Gupta, “Path Planning with General End-effector Constraints,” Robotics and Autonomous Systems, Vol. 55, pp. 316-327, 2007.

[85] Md. Nazrul Islam, Shinsuke Tamura, Tomoya Murata and Tatsuro Yanase “Evaluation of a New Backtrack Free Path Planning Algorithm for Manipulators," IEEJ Trans. EIS, Vol. 128, No. 8, pp. 1293-1302, 2008.

[86] S. Tamura, T. Murata, Md. Nazrul Islam, T. Yanase, and S. Taniguchi “A Path Planning Algorithm for Multi Manipulators”, IECON' 08 under review.

Author Bibliography

Name: Md. Nazrul Islam

Permanent Address: Village: Polash Bari; Post Office: Gokul; District: Bogra-5800; Bangladesh

E-Mail: nazrul.02@gmail.com

Nationality: Bangladeshi

Education:

2008 Ph.D. in information Science, University of Fukui, JAPAN

1998 M.phil in Applied Mathematics, Bangladesh University of Engineering & Technology

1995 M.Sc. in Applied Mathematics, Dhaka University, Bangladesh

1993 B.Sc. in Applied Mathematics, Dhaka University, Bangladesh