

Manipulator Path Planning by Decomposition: Algorithm and Analysis

Arjang Hourtash and Mahmoud Tarokh, *Senior Member, IEEE*

Abstract—Path planning is achieved by a special decomposition of the robot manipulator, an off-line preprocessing stage, and a three phase on-line path planning scheme. The decomposition consists of disassembling the robot into several chains where a chain is a combination of several consecutive links and joints. Preprocessing is performed by defining a set of postures for each chain, and setting up a collision table which re-integrates the chains into the full robot and stores the collision states of various discretized robot configurations with the obstacles. Path planning using a local search is performed independently in joint subspaces associated with robot chains. The paths found for the chains are synthesized to obtain a collision-free path for the robot. This decomposition reduces the exponential growth of computation with robot degrees of freedom (DOF) to that of the much lower chain DOF. As a result it is possible to achieve short planning times for practical robots operating in 3-dimensional work spaces. Analysis of computation time and space of the proposed method are presented. Results supporting the analysis are provided for a large number of path planning trials with two practical robots operating in relatively cluttered environments.

Keywords—Path planning, Manipulator Decomposition.

I. INTRODUCTION

MOTION planning is one of the most fundamental problems in robotics. The objective of motion planning is to break down a high level task into low level actuator commands. An important aspect of motion planning is path planning, which addresses the geometric concerns of robot motion without regard to time [21]. In this paper, we address the problem of moving a set of actuated and constrained robot links from a start configuration to a goal configuration in a static obstacle environment.

Path planning is often formulated by transforming the work space volume occupied by the robot into a single vector or point in the robot configuration space or C-space [3], [23], [24], [25]. The work space obstacles are transformed into the forbidden regions of the C-space. As a result, a collision free robot path is a curve which circumvents the forbidden regions in the C-space. An n -DOF manipulator has an associated n -dimensional C-space. High dimensional C-spaces cause rapid growths in computation required for modeling and planning [4]. The reduction of computation effort associated with path planning is the focus of this paper [11]-[12], [28]-[31].

In the past two decades, many local and global approaches to path planning have been presented. Local planners construct the path incrementally, based on the subset

of the environment in the vicinity of the leading edge of path construction. Therefore they are often fast, but lack the necessary broad perspective, and can often get stuck in traps. Global planners conceive their paths after a complete survey of the environment, and are therefore much slower, but are generally much more trap resistant than local planners.

The most common strategies for path planning are road map, cell decomposition, and potential field methods [21]. Road map methods generate a connected network of one-dimensional curves called roads which capture the connectivity of the C-space. During path planning the start and goal are connected to this road map which is searched for a path. Cell decomposition methods discretize the C-space into cells, leading to a graph of either “clear” or “in-collision” cells [32]. Path planning then consists of determining the cells containing the start and goal configurations, followed by searching the graph for a clear path to connect these cells. A variant of the cell decomposition method is employed in [5] and [15] which breaks up the C-space into coarse cells, and maintains a connectivity map of these cells. A search is used to plan paths between the coarse cells, and these cells may be hierarchically decomposed. Potential field methods model the C-space by a potential function, where the global minimum is assigned to the goal, and high potentials to the obstacles. Such a potential function serves to create perceived forces to repel the robot from the obstacles while attracting it toward the goal [19], [20], [27]. Path planning then consists of traversing down the surface of the potential function to the goal. Potential functions are best known for their real-time control applications and their susceptibility to local minima.

These basic strategies have been modified in the past decade creating the randomized methods among others, which have made faster path planning possible [2]. Probabilistic road map methods [17]-[18] model the environment off-line by a random set of collision free configurations generated in the learning phase according to the local complexity of the environment. During path planning the start and goal are connected to the road map to complete the path. If connecting either the start or goal to the road map fails, then random configurations are generated in the vicinity of the problematic end-point to ultimately connect it to the road map. This approach requires frequent on-line collision detections which are computationally intensive for practical manipulators in 3-dimensional work spaces. Another approach based on randomization [1], constructs a potential function using on a skeleton of the work space, that represents the distance to the goal location of the end-effector. Using a number of control points assigned to the

A. Hourtash is with Simplify Robotics, Inc., 15738 Pipers View Dr., Webster, TX 77598. Email: hourtash@alumni.ucsd.edu

M. Tarokh is with the Intelligent Machines and Systems Laboratory, Department of Computer Science, San Diego State University, San Diego, CA 92182-7720. Email: tarokh@sdsu.edu

manipulator body, this work space potential function is mapped into a potential function in the C-space. At each step during path planning, a neighbor of the leading edge of the path with a lower potential is chosen for the next step. In a high dimensional C-space, the number of neighbors is large, and thus only a small randomly chosen subset of the neighbors are examined. If none of the examined neighbors have lower potentials, a random walk mimicking Brownian motion is initiated. This random walk combined with the generation of the skeleton for a 3-dimensional work space may require relatively large computation times.

In order to reduce the planning time, a sequential search strategy is proposed in [8]-[10]. Each search plans the motion for one link based on the resultant paths of the previous links. This requires the construction of certain two dimensional product spaces. Collision detection is then performed to build 2-dimensional bitmaps, and the product spaces are searched for a path. The strength of this method is that the n -dimensional environment is represented using a number of 2-dimensional bitmaps, which substantially reduces computation space. Its weakness is that the generated bitmaps involve on-line collision detection, making this strategy computationally slow.

The present paper addresses the reduction of the computations associated with path planning. The proposed approach uses two main principles. First, the robot and environment are modeled off-line, and the results are stored for repeated use by the on-line planner. This eliminates many unnecessary and repetitive on-line computations. Second, instead of planning the path for the manipulator as a whole, the manipulator is decomposed into a number of *chains* of links whose paths are planned independently. The results from all chain paths are then combined to construct the path for the whole manipulator. This unique decomposition reduces the n dimensional robot path problem into m sets of n_i -dimensional chain path problems, where m is the number of chains, n_i is the DOF of the i -th chain and $n = \sum_{i=1}^m n_i$. As a result, the exponential growth of computation with robot DOF [4] is reduced to exponential growth with $\max_i(n_i, m)$. This allows very short planning times that have previously been unattainable.

II. OFF-LINE MODELING

Off-line modeling consists of generating a set of discretized configurations whose collision states with the known and static environments are determined and stored in a table for use by the on-line planner. The concept of robot decomposition plays a central role in both modeling and planning, and is discussed in the following section.

A. Robot Decomposition

In order to speed up path planning, an n DOF manipulator is decomposed into *chains*, where a chain refers to a set of consecutive links and their associated joints. In a manipulator, chains could be torso, arm, and wrist. An example of such a decomposition is shown in Fig. 1. The 5-link 7-DOF manipulator in this figure is decomposed into three chains.

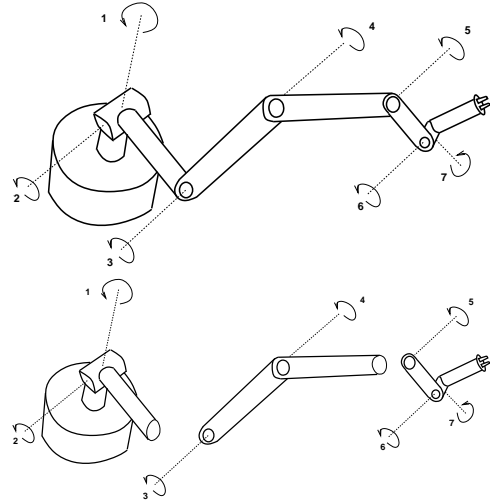


Fig. 1. Decomposition of a 5-link 7-DOF manipulator into 3 chains.

In general, the $n \times 1$ robot joint variable vector Θ is decomposed into its component vectors Θ_i , $i \in \{1, \dots, m\}$, where Θ_i is the $n_i \times 1$ joint variable vector associated with the i -th chain and m is the number of chains. The assignment of particular links and their associated joints to the chains will be referred to as the robot *decomposition style* \mathcal{D} . For a given n -DOF robot, there may be many possible decomposition styles. However, for practical and effective planning, these decomposition styles must satisfy the following four permissibility requirements: 1) The DOF of a link must not be decomposed, i.e. all joint variables of a compound (multi-DOF) joint must be assigned to the same chain. 2) The links and joints of a chain must form a consecutive sequence. 3) The sum of the chain DOF must add up to the robot DOF n , that is $\sum_{i=1}^m n_i = n$, where n_i is the number of DOF of chain ξ_i . 4) Each chain must have at least two DOF, that is $n_i \geq 2$. The last requirement eliminates the possibility of degenerate chains with only 1 DOF. Such a 1 DOF chain does not have sufficient manipulability for collision avoidance, since it only allows forward and reverse motions about its joint axis. Based on these requirements, the number of chains m must satisfy

$$2 \leq m \leq \frac{n}{2} \quad (1)$$

Consider the 7-DOF 5-link robot in Fig. 1. The first link (closest to the base) $L_1(\theta_1, \theta_2)$ is actuated by a compound joint with joint angles θ_1 and θ_2 . The next links $L_2(\theta_3)$, $L_3(\theta_4)$ and $L_4(\theta_5)$ are actuated by simple joints. Finally the last link $L_5(\theta_6, \theta_7)$ also has a compound joint. In the following, the abbreviated notation $\mathcal{D} = \{n_1 \dots n_m\}$ showing only the DOF of the chains, will be used to express decomposition styles. In this notation, the decomposition style $\mathcal{D} = \{34\}$ indicates that the first chain has 3 DOF, a joint variable vector $\Theta_1 = (\theta_1, \theta_2, \theta_3)$, and consists of links $L_1(\theta_1, \theta_2)$ and $L_2(\theta_3)$. The second chain has the remaining 4 DOF, a joint variable vector of $\Theta_2 = (\theta_4, \theta_5, \theta_6, \theta_7)$, and consists of links $L_3(\theta_4)$, $L_4(\theta_5)$ and $L_5(\theta_6, \theta_7)$. Examples of

permissible two and three chain decompositions for the manipulator of Fig. 1 are, $\mathcal{D} = \{223\}$ and $\mathcal{D} = \{34\}$. On the other hand, a decomposition style such as $\mathcal{D} = \{1213\}$ distributes the DOF of the link $L_1(\theta_1, \theta_2)$ among two chains, and the first chain has only one DOF. Thus, it fails the requirements 1 and 4. Among many permissible decomposition styles, some provide more efficient path planning, as described in Sections IV and V.

A.1. Chain Postures – The first task of the off-line modeling is to represent the range of motion of the chains. A chain ξ_i can assume an infinite number of *postures* depending on the value of its continuous joint variable vector Θ_i . Each of these postures represents a formation of the links and their volume in the work space. For computer implementation, the continuum of robot motion is discretized into a finite number N_i of Θ_i values denoted as $\Theta_{i,j}$, $i \in \{1, \dots, m\}$, $j \in \{1, \dots, N_i\}$. The corresponding *discretized chain postures* are denoted as $P_{i,j}$. The combination of one posture from each of the m chains constitutes a *discretized robot configuration* C_k , that is

$$C_k = (P_{1,j_1}, \dots, P_{m,j_m}) \quad (2)$$

There are N discretized configurations, $C_k, k \in \{1, \dots, N\}$, where $N = \prod_{i=1}^m N_i$. The discretized postures and configurations, populate the joint subspaces of the chains and the joint space of the robot, respectively. Robot paths will be represented by sequences of discretized configurations.

A.2. Separation and Resolution – In order to delicately maneuver the robot in a cluttered environment, the discretized configurations must be densely distributed. Dense distributions lead to higher computation costs. So, a sparsely populated environment with bulky objects calls for fewer discretized configurations. As a result, configurations must be generated with their separations tailored to the environment. The *separation* $S(C_1, C_2)$ is a measure of spacing in the work space between the physical links of the robot at two different configurations C_1 and C_2 , and is defined more precisely as follows.

Suppose that at a given configuration, a point $q(t, k)$ is free to slide along the centerline \mathcal{Q} of the robot; where the centerline \mathcal{Q} is defined as a set of line segments representing and connecting the axes of the links and joints; and t is a parameter indicative of the position along \mathcal{Q} . Here the base of the robot is identified as $q(0, k)$, and the end-effector as $q(1, k)$. The second argument of $q(t, k)$ specifies the configuration C_k of the robot by its index k . The separation measure $S(\cdot, \cdot)$ between two configurations C_1 and C_2 is defined as

$$S(C_1, C_2) = \max_t \|q(t, 1) - q(t, 2)\|_2, \quad q(t, \cdot) \in \mathcal{Q} \quad (3)$$

where $q(t, 1)$ and $q(t, 2)$ are on the centerline \mathcal{Q} of the robot as it assumes the configurations C_1 and C_2 , respectively.

Physically, (3) represents the largest distance in the work space that any point on the centerline \mathcal{Q} travels as the

robot moves from C_1 to C_2 . Fig. 2 illustrates a 6 DOF 4-link robot, with its points $q(t, 1)$ and $q(t, 2)$ free to slide along the centerline. The maximum Euclidean distance between $q(t, 1)$ and $q(t, 2)$ for $t \in [0, 1]$ is labeled as the separation.

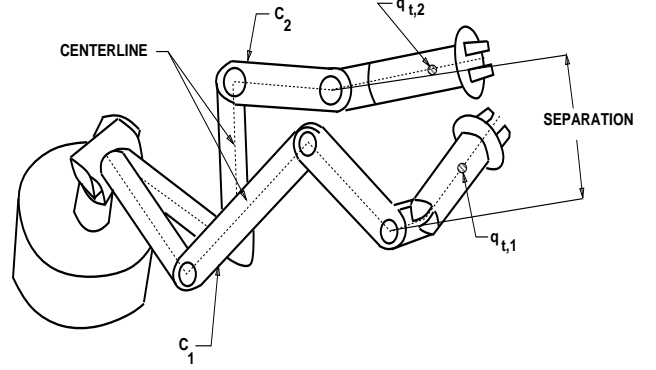


Fig. 2. Separation between two configurations.

Since the robot is decomposed into chains whose postures are used for path planning, separation must also be defined for postures. Posture separation is defined as the maximum work space distance that a point on the centerline \mathcal{Q} travels as chain ξ_i moves from $P_{i,1}$ to $P_{i,2}$ while the postures of all other chains $\xi_j, j \neq i$ are set to be identical and to maximize the distance. Mathematically,

$$S(P_{i,1}, P_{i,2}) = \max_{P_{j,*}, j \neq i} \left(\max_t \|q(t, 1) - q(t, 2)\|_2 \right) \quad q(t, \cdot) \in \mathcal{Q} \quad (4)$$

where $P_{j,*}$ under the outermost maximization has replaced $\{P_{j,1} = P_{j,2}\}$ for brevity. In [14], it is shown that both configuration separation and posture separation are metrics.

Two postures $P_{i,1}$ and $P_{i,2}$ are said to be *adjacent* if their separation satisfies

$$S(P_{i,1}, P_{i,2}) \leq S_r \quad (5)$$

where S_r , is the *resolution*, and is selected according to the robot's interaction with the obstacles. A coarse resolution (large S_r) leads to unnecessarily gross approximations of the robot motion, and reduces path planning success. On the other hand, a resolution chosen to be too fine (small S_r) yields an unnecessarily large number of discretized postures N_i , which wastes computation time and space. For these reasons, the resolution S_r must be chosen as a balance between planning success and efficiency. The resolution S_r is normalized with respect to the length l_1 of the fully extended manipulator from the base to the tip of the end-effector, to yield the *normalized resolution* R , that is

$$S_r = R l_1 \quad (6)$$

A.3. Coverage and Connectivity – In addition to resolution, two properties, namely *coverage* and *connectivity*, describe the quality of the discretized posture population. This stems from the fact that in a graph-based path planning

such as this, the two main objectives are 1) connect arbitrarily specified start and goal configurations to the graph of discretized configurations, and 2) search for a path in the graph to connect the start and goal. Coverage ϕ_i of the chain ξ_i quantifies the first objective, i.e. how well the discretized postures are distributed so that they can represent any arbitrary non-discretized posture. Section III describes robot paths as sequences of collision free discretized configurations. Motion between these discretized configurations is also collision free if the collision checking is performed with the robot's link shrouds expanded by $\frac{S_r}{2}$ normal to shroud surfaces [14]. This allows for each discretized posture $P_{i,j}$ to represent a continuum of postures, referred to here as the *spanning region* of the discretized posture. For a given resolution S_r , an arbitrary posture $P_{i,a}$ is representable by a discretized posture $P_{i,j}$ if $P_{i,a}$ lies in the spanning region of $P_{i,j}$, or alternatively if

$$S(P_{i,a}, P_{i,j}) \leq \frac{S_r}{2} \quad (7)$$

Coverage is satisfactory if any arbitrary posture has at least one *nearby* discretized posture where nearby is defined by (7). Coverage ϕ_i of the chain ξ_i is measured by randomly creating a large enough number $N_{i,a}$ of arbitrary postures which are tested for nearby discretized ones. Those which have at least one nearby discretized posture are counted as $\hat{N}_{i,a}$. Coverage is now quantified as

$$\phi_i = \lim_{N_{i,a} \rightarrow \infty} \left(\frac{\hat{N}_{i,a}}{N_{i,a}} \right)^2 \quad (8)$$

and measures the likelihood that both start and goal have nearby discretized postures. The squaring accounts for both the start and goal. Coverage ranges as $\phi_i \in (0, 1]$ where $\phi_i = 0$ corresponds to no arbitrary postures having a nearby discretized postures for chain ξ_i , and $\phi_i = 1$ occurs when every arbitrary posture has one or more nearby discretized postures.

The second property, the *connectivity* ψ_i of chain ξ_i , quantifies the second objective, namely how well the discretized postures are connected to each other, or alternatively the likelihood that an arbitrary pair of discretized postures $P_{i,1}$ and $P_{i,2}$, are connectable via a path of consecutively adjacent postures, where adjacency is defined by (5). Suppose that the graph of discretized postures for the chain ξ_i consists of $N_{i,conn}$ connected subgraphs. Then connectivity is defined as

$$\psi_i = \sum_{k=1}^{N_{i,conn}} \left(\frac{N_{i,k}}{N_i} \right)^2 \quad (9)$$

where $N_{i,k}$ is the number of discretized postures in the k -th connected subgraph. The squaring of the terms addresses the fact that both the start and goal must be in the same connected subgraph. In a fully connected graph $N_{i,conn} = 1$ and $N_{i,k} = N_i$, giving $\phi_i = 1$. On the other hand, in a totally disconnected graph, $N_{i,conn} = N_i$ and $N_{i,k} = 1$, resulting in a connectivity of $\psi_i = \frac{1}{N_i} \approx 0$, since

N_i is generally large. High coverage and connectivity are required for successful planning.

A.4. Generation of Discretized Postures – The posture generation method must be adjustable to yield the desired resolution, coverage, and connectivity. Posture generation involves two steps: candidate posture generation, followed by posture acceptance.

Initially, a set of *candidate* postures is generated for each chain. This is done by dividing the range of angular motion of each DOF of the joints into small divisions $\Delta\theta_j$ based on the desired normalized resolution R . Consider two configurations that differ only in a single joint, say joint j , by the value $\Delta\theta_j$. Let the desired separation at the end-effector be S_r given by (6). The angle difference $\Delta\theta_j$ forms an isosceles triangle whose two equilateral sides are each l_j long, namely the length from joint j to the end-effector, and its base is $S_r = Rl_1$. Simple trigonometry gives $\Delta\theta_j = 2 \sin^{-1} \left(\frac{Rl_1}{2l_j} \right)$. We further reduce this angle difference by introducing a *generation coefficient* $K_g \in (0, 1]$ to get

$$\Delta\theta_j = 2K_g \sin^{-1} \left(\frac{Rl_1}{2l_j} \right) \quad j = 1, 2, \dots, n \quad (10)$$

The particular form chosen for $\Delta\theta_j$ above ensures that the basal joints (those closer to the base) have smaller increments than joints closer to the tip since small changes in basal joints cause larger manipulator motion in the work space. Furthermore, fine resolutions (small R), as well as small K_g , lead to small joint angle increments $\Delta\theta_j$. As a results, the generation coefficient K_g provides a way for reducing joint angle increments without changing the resolution. These joint angle divisions then combine to form a rectanguloid grid of candidate postures in the joint subspace of the respective chain ξ_i .

Due to the complex mapping from the C-space to the work space, certain regions of the work space may be over-represented by the candidate postures. Thus, many candidates can be removed without degrading either the coverage or the connectivity. Planning with fewer discretized postures having the same coverage and connectivity, results in less computation time and space without reduction in planning success. Therefore, the usefulness of the candidate postures are examined using the following acceptance criterion, and only those candidates passing this criterion become the discretized postures. This criterion requires the candidate posture being examined to have a separation greater than $K_a S_r$ from every discretized posture accepted up to that time. Here K_a is the *acceptance coefficient*, and is bounded as $K_a \in [0, 1]$. Essentially this acceptance criterion rejects the unnecessary candidate postures to ensure that no two discretized postures are within a separation of $K_a S_r$. In this way, K_a sets the degree to which similar candidates are to be rejected.

Note that the above method of pruning is different from that of placing a regular grid in the chain subspace and selecting a subset of the adjacencies, e.g. grid nodes that differ only in one coordinate. That would preclude diagonal

steps in a chain subspace which may limit planning success, and lengthens the path. On the other hand, pruning by the proposed method selects each adjacency from the set of all N_i postures without this limitation.

For each discretized posture $P_{i,j}$, an adjacency list is constructed that stores those postures of the chain ξ_i that are within a separation of S_r from $P_{i,j}$. Since separation of each posture from all other $(N_i - 1)$ postures must be computed, this requires $O(N_i^2)$ separation evaluations each involving several homogeneous transformations, and is thus computationally intensive. Here $O(\cdot)$ is the order notation. Appendix A describes a computationally effective exclusion test to reduce the number of required separation evaluations from $O(N_i^2)$ to $O(N_i n_i)$, where $n_i \ll N_i$.

B. Environment Modeling

The purpose of decomposing the robot into chains is to reduce the dimensionality of the overall problem which in turn reduces the computation time and space of planning. However these chains must be recombined in order to determine the interaction of the manipulator with the environment. The off-line modeling of these interactions is now discussed.

The robot-obstacle interactions are modeled by assigning one of two possible collision states, “clear” or “in-collision”, to discretized configurations C_k . The collision states are determined by first expanding the shrouds, followed by pruning, and finally standard geometrical interference checking [4], [7], [22], [26].

B.1. Collision Table – The results of the off-line collision checks are stored in a collision table. This table can then be queried during on-line path planning to ensure obstacle avoidance. The collision table is an m -dimensional, or m -axis, array whose i -th axis consists of N_i divisions, representing the N_i discretized postures of chain ξ_i . Thus from (2), the cells represent the discretized configurations of the robot. Since the i -th axis of the collision table is a 1-dimensional representation of chain ξ_i , which has an n_i -dimensional joint subspace, the postures cannot be ordered based on any metric. Therefore, the discretized configurations represented by neighboring cells are *not* necessarily adjacent in the work space. Consequently, adjacencies between cells are determined based on the posture adjacency lists that were set up during posture generation. During collision checking, the resulting collision states of the discretized configurations are recorded in the appropriate collision table cells.

The collision table is a compressed representation of the C-space. In the extreme case where the acceptance coefficient $K_a = 0$ thus accepting all candidate postures, the size (number of entries) of the collision table is exponential in the robot DOF. On the other hand, when $K_a = 1$ thus rejecting all candidate postures, the collision table has no entries. Optimal values for K_a are less than 0.5, as will be discussed in Section IV-C and as a result, the selective acceptance of the candidate postures substantially reduces the size of the collision table. This off-line collision table

creation improves the on-line planning time. However, the main contributor to efficient planning is the decomposition and a three phase planning procedure, as will be discussed in Section III.

B.2. Collision Probability – Path planning is performed for each chain independently of the other chains. However, collision states can only be determined for configurations rather than for postures. Nevertheless, the probability that a posture $P_{i,j}$ is involved in collision can be assessed. Since there are N discretized configurations and N_i discretized postures for each chain ξ_i , the number of discretized configurations that have $P_{i,j}$ as one of their component postures is $\frac{N}{N_i}$. Let $\mu_{i,j}$ denote the number of in-collision configurations associated with the posture $P_{i,j}$. Then the *collision probability* of the posture $P_{i,j}$ is

$$\gamma_i(P_{i,j}) = \frac{\mu_{i,j}}{(N/N_i)} \quad (11)$$

The collision probability of each posture is evaluated using the collision table during the off-line stage and is stored for on-line path planning.

The time taken to perform preprocessing, including the generation of discretized postures and setting up the collision table, can be fairly long if the resolution is chosen too fine, or if the decomposition style is not chosen properly. The most intensive operations during preprocessing are separation evaluation and polyhedral interface checking for collision detection. If the design parameters are chosen according to the procedure described later in Section IV-C, preprocessing on a reasonably cluttered environment can take a few hours on the same machine (e.g. a 333 MHz PC) that would do the on-line planning in a few hundredth of a second. However, once completed, the results can be used for an infinite number of path planning for the same robot and environment. Therefore, the preprocessing time is very small, if amortized over a large number of path planning trials. It is also noted that preprocessing can be partially done in parallel to reduce the preprocessing time. In this case, processes working on posture generation of different chains may work independently and in parallel. The collision detection may also work independently if each process is assigned a set of configurations. Parallel preprocessing has not yet been implemented, and is currently being investigated.

III. ON-LINE PLANNING

The path planner employs the robot and environment models developed in Section II to plan a collision free path in three phases. Phase 1 finds a pair of discretized postures from each of the m chains to represent the actual start and goal configurations. Phase 2 connects the start and goal postures for each chain. Phase 3 synthesizes the resulting m chain postures into a final robot path.

A. Phase 1: Identifying the Discretized Start and Goal Configurations

In general, the actual start and goal do not exactly coincide with any discretized configurations. Therefore, a connection must be made between the actual start \tilde{C}_s and the discretized start C_s , as well as one between the actual goal \tilde{C}_g and the discretized goal C_g configurations. To find a discretized start configuration C_s , the actual start configuration \tilde{C}_s is first decomposed into its m component actual start postures $\tilde{P}_{i,s}$, $i \in \{1, \dots, m\}$. Then, each of the m chains are searched for discretized postures within a separation of $\frac{S_r}{2}$ from $\tilde{P}_{i,s}$. The selected discretized postures are combined, one for each chain, to form a candidate for the discretized start C_s configuration. In order for the discretized start C_s to represent the actual start \tilde{C}_s , C_s must both be clear from collision, and be in the spanning region of \tilde{C}_s . Otherwise the process of selecting $P_{i,s}$, reconstructing into a candidate C_s , and checking for being collision free and nearby the actual \tilde{C}_s is exhaustively repeated until C_s is identified. Note that if the coverage is satisfactory, such a C_s exists. This search is repeated for the discretized goal configuration C_g .

B. Phase 2: Constructing Independent Chain Paths

Phase 2 employs a local graph search to construct independent paths connecting the discretized start and goal postures for each of the m chains. This connection is made by iteratively constructing a sequence of consecutively adjacent discretized postures for each chain. Each iteration picks an adjacency of the most recently selected discretized posture and adds it to the sequence. This selection is based on minimizing an objective function which incorporates goal seeking and obstacle avoidance as follows.

For the goal seeking, the arc-length $\mathcal{L}(\Theta_{i,1}, \Theta_{i,2})$ is used which is the Euclidean distance in joint space between the joint variable vectors of postures $P_{i,1}$ and $P_{i,2}$, with a modification in the case a joint has no mechanical limits. In this case, the arc-length returns the length of the geodesic line connecting its two arguments, $\Theta_{i,1}$ and $\Theta_{i,2}$, in the multiply connected joint space. The arc-length is used instead of the separation, because it is measured in joint space rather than in work space, which leads to smoother joint paths. The collision probability $\gamma_i(\cdot)$ discussed in the Section II, is used for the obstacle avoidance component of the objective function.

At each iteration, the selection of the next discretized posture for the path is based on the minimization of the objective function Q_i over all adjacencies $P_{i,j}^a$ of $P_{i,j}$,

$$\min_{P_{i,j}^a} Q_i(P_{i,j}^a) \quad (12)$$

where $P_{i,j}$ is the most recently selected discretized posture during the current iteration. A simplified objective function is

$$Q_i(P_{i,j}^a) = \frac{\mathcal{L}(\Theta_{i,j}^a, \Theta_{i,g})}{1 - \gamma_i(P_{i,j}^a)} \quad (13)$$

where $\Theta_{i,j}^a$ is the joint variable vector associated with an adjacent posture $P_{i,j}^a$, and $\Theta_{i,g}$ is that for the goal posture $P_{i,g}$. The numerator is intended for goal seeking and the denominator uses $\gamma_i(P_{i,j}^a) \in [0, 1]$ to discourage the selection of a posture with a high collision probability. As each posture is selected, it is marked to prevent selecting it again which would result in periodic paths. In general a dead end is possible, and is encountered when the planner wraps itself into a trap with all its adjacent postures either having been previously visited, or having a collision probability of one. Phase 2 is successfully completed when the goal is reached for all m chains; otherwise failure is declared if any chain encounters a dead-end.

In conclusion, Phase 2 path planning maps the n_i dimensions of the joint subspace of ξ_i onto a 1-dimensional path. This is a dimensional reduction of $(n_i - 1)$, which is useful only when $n_i \geq 2$. This further justifies the fourth permissibility requirement for decomposition discussed in Section II-A. The result of Phase 2 is a sequence of ρ_i consecutively adjacent discretized postures, for each of the m chains.

C. Phase 3: Synthesizing the Robot Path

Since definitive collision checking cannot be performed for postures, the m chain paths from Phase 2 do not fully represent the interactions of the whole robot with the obstacles. Therefore in Phase 3, these m chain paths are synthesized into the final robot path, by combining the utilities of the collision table and adjacency lists into the following paradigm of the *work table*.

The work table is similar to the collision table and has m axes, each representing a chain. However, its i -th axis has ρ_i divisions representing the sequence of consecutively adjacent discretized postures of the chain ξ_i path selected in Phase 2. Thus the size of the work table is $\prod_{i=1}^m \rho_i$. The work table differs from the collision table whose axes represent the sets of all N_i discretized postures and whose size is $\prod_{i=1}^m N_i$, where in general $N_i \gg \rho_i$. In contrast to the collision table, non-diagonal neighboring cells of the work table are adjacent since the postures along its axes are consecutively adjacent. The collision states of the cells are read directly from the collision table. Since the start posture $P_{i,s}$ and the goal posture $P_{i,g}$ are at the ends of the chain paths, the start and goal of the robot path are always a pair of cells on diametrically opposed corners of the work table.

The relatively small size of the work table enables fast implementation of a global search such as breadth-first or depth first graph search. These global search methods are immune to local difficulties such as local traps or dead-ends, and will always find a path in the work table, if it exists. The result of a successful search is a sequence of clear, consecutively adjacent configurations joining the discretized start and goal. The search is unsuccessful if the work table is disconnected, which is possible even if Phase 2 successfully completes all m chain paths.

D. Advanced Strategies

The three phase path planner described thus far is vulnerable to a number of difficulties. The following discusses some limitations of this planner, and some heuristic approaches to address many of them. These topics include path conditioning, backtracking, and retrograde motion.

D.1. Path Conditioning – A complicated chain path may consist of twists and turns. This situation can lead to two general problems: 1) The overall path becomes encumbered with inadvertent and irregular motions. 2) The work table becomes excessively large and complicated, which in highly cluttered regions, may lead to its disconnectedness and to Phase 3 failure. For this reason, the chain paths of Phase 2 are conditioned before starting Phase 3. This entails performing a breadth-first search on each of the m chain paths, individually. For a given chain ξ_i path, the ρ_i postures on the path become the graph vertices, and all possible adjacency relationships between them, become the graph edges. The breadth-first search leads to a simple chain path from the start posture $P_{i,s}$ to the goal posture $P_{i,g}$. Path conditioning also includes provisions for giving preference to postures with the least collision probabilities γ_i .

As an illustration of path conditioning, consider the posture sequence $\{P_{1,1}, P_{1,2}, P_{1,7}, P_{1,34}, P_{1,9}\}$ representing the chain ξ_1 path constructed during Phase 2. Suppose further that both $P_{1,7}$ and $P_{1,9}$ are adjacent to $P_{1,2}$. Path conditioning bypasses $P_{1,7}$ and $P_{1,34}$ to produce $\{P_{1,1}, P_{1,2}, P_{1,9}\}$. The jaggedness in the original path is caused by the conflicts between multiple criteria, i.e. goal seeking and obstacle avoidance. As a result, the conditioning leaves already simple paths unchanged, while complicated and twisted paths that previously folded onto themselves are greatly simplified. This leads to smoother and more direct overall robot motion, as well as fewer work table cells which improves planning success rates in high collision probability regions. However caution must be exercised, since in certain scenarios the tight turns necessary for circumventing obstacles may also be eliminated by this conditioning.

D.2. Backtracking – Although the planner is demonstrated to have reasonable success rates, as will be seen in Section V, an existing path at a given resolution may still elude the planner. This is an attribute of all planners employing local search techniques such as the one used in Phase 2. Note that using a global search in Phase 2 would not alleviate this problem and would only slow the planner, since definitive collision checking cannot be associated with independent postures. Thus, a backtracking algorithm is presented to improve the success of phases 2 and 3.

Recall that Phase 2 incrementally constructs the sequence $\{P_{i,s}, \dots, P_{i,g}\}$ to represent the path for chain ξ_i . Phase 2 fails if all the adjacencies of the most recently selected posture have either been previously selected and marked, or have a collision probability of one. In this case,

the most recent addition to the path is denoted as the *failure posture* $P_{i,f}$. Examination of collision probability γ_i surfaces has revealed that these surfaces can in general be extremely complicated, and the direction of path construction, i.e. start-to-goal or goal-to-start, is often an important consideration. If the start-to-goal construction faces a steep uphill in the collision probability surface, then planning may fail. The downhill goal-to-start, however, can lead to a successful path.

Therefore, to improve the success when a failure posture is encountered, Phase 2 backtracking reverses the path construction direction. In addition, the partially completed path is utilized by treating *any* of its postures as a possible target. For illustration consider the incomplete sequence $\{P_{i,s}, P_{i,1}, P_{i,2}, P_{i,f}\}$. Since a failure posture $P_{i,f}$ is detected, the path construction is continued by reversing the path construction direction. The original goal posture is now treated as the new start posture, and the path construction attempts to reach any of the postures $P_{i,s}$, $P_{i,1}$, or $P_{i,2}$. Note that the problematic areas may again cause failures. Thus in the example, we may get the reverse sequence $\{P_{i,g}, P_{i,10}, P_{i,9}, P_{i,8}, P_{i,f1}\}$. In this case $P_{i,f1}$ is marked, and the planner backs up a random number of postures, e.g. to $P_{i,9}$, and tries to reach any of the postures $P_{i,s}$, $P_{i,1}$, or $P_{i,2}$. If necessary, this backtracking continues until it succeeds, or it backs up all the way to $P_{i,g}$, in which case a total failure is declared.

The failure in Phase 3 planning is due to the start and goal cells being disconnected in the work table. In this case the closest cell to the goal that is visited by the breadth-first search is labeled as the failure configuration C_f . Phase 2 backtracking then reconstructs the desired chain paths, with $P_{i,f}$ defined as the component postures of C_f . Repeating Phase 3 then combines the reconstructed chain paths. In essence, this Phase 3 backtracking reuses the successful portion, and reconstructs the remainder of the work table.

The concept of backtracking, i.e. the reconstruction of a failed path in the vicinity of a failure, has also been employed in [9]. In response to a failure, the algorithm in [9] moves back to the previous link, inserts a virtual forbidden regions in the 2D parametric space of that link in an attempt to divert the planner away from the trouble region, and then planning is redone from that link. In the proposed method when a failure is encountered in Phase 2, planning is reinitialized from the goal towards *any* posture on the partially constructed path. This multiple target policy considerably improves the chances of path finding.

The backtracking slows the planning, however it improves the success rate substantially. Nevertheless despite backtracking, it is still possible that an existing path at a given resolution may elude the planner.

D.3. Retrograde Motion – The decomposition of the robot into chains may in some cases destroy the n -dimensional “big picture” perspective which is necessary for path construction. Fig. 3 shows a very simple example in four frames from left to right for a 2 DOF manipulator. Although decomposition is unnecessary for such a simple 2

DOF manipulator, the example will demonstrate the concept. In this example, first the basal link of the 2-DOF manipulator is moved away from the obstacle to make room for its distal link. Then, both links are moved toward the target. We refer to this coordinated back and forth motion as *retrograde motion*. If the DOF of these two links are isolated from each other through decomposition, the necessary coordination and hence the back and forth motion would not be possible. The path construction discussed in Section III-B can be modified to shift the emphasis between goal seeking and obstacle avoiding based on the fraction of the path completed. For example, in the beginning of a path, the emphasis can be placed on obstacle avoidance which can result in path construction moving away from the goal if obstacles are encountered. On the other hand near the goal, path construction can be based heavily on goal seeking which causes the shortest connection to the goal. This retrograde motion allows for the back and forth motion which generally improves the chances of success. However, the most effective remedy to the loss of big picture perspective is usually proper selection of the decomposition style which will be discussed in Section IV-C.

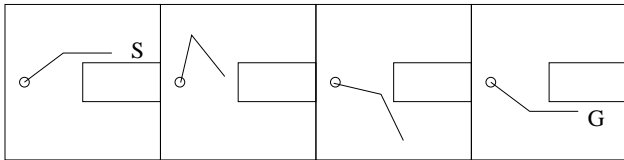


Fig. 3. Back and forth link motion of basal link.

IV. ANALYSIS

This section presents the computation analysis of the proposed planner and demonstrates the advantages of the decomposition. The focus of the analysis is on the basic planning method, and thus the advanced strategies in Section III-D are not considered in the analysis. Furthermore, the emphasis of this section is on identifying the major contributors to the computation effort, and for this purpose we use order notation to succinctly express appropriate relationships. Appendix B presents explicit equations for the theoretical computation time.

A. Computation Time

A.1. Phase 1 – The computations in this phase involve finding a pair of discretized start and goal postures $P_{i,s}$ and $P_{i,g}$ that are nearby the actual start and goal $\tilde{P}_{i,s}$ and $\tilde{P}_{i,g}$, for each of the m chains, as defined by (7). The $\tilde{P}_{i,s}$ and $\tilde{P}_{i,g}$ result from decomposing the actual start and goal configurations \tilde{C}_s and \tilde{C}_g . For brevity, the following discuss the start of the path. The same arguments apply to the goal. The exclusion test described in Appendix A is applied to the discretized postures of the m chains to find neighbors of the actual start $\tilde{P}_{i,s}$ postures. For each chain, an estimated $O(N_i)$ postures are examined using

the exclusion test; where N_i is the number of discretized postures. For each exclusion test, an $O(n_i)$ mathematical operations (*ops*) are performed where n_i is the DOF of the i -th chain. Therefore, the computation time of the exclusions of all chains is $O(\sum_{i=1}^m N_i n_i)$ ops.

As the discretized postures from each of the m chains pass the exclusion tests, they are combined to form a discretized start configuration. There are two requirements for the actual start \tilde{C}_s to be represented by a discretized configuration C_s . First, C_s must be clear from collision, and second it must be in the spanning region of \tilde{C}_s , i.e., $S(\tilde{C}_s, C_s) \leq \frac{S_r}{2}$. These require collision table look-ups and separation evaluations, respectively, and are done in $O(n^2)$, [13]. The computation time of Phase 1 is then

$$T_1 = O\left(\sum_{i=1}^m N_i n_i\right) + O(n^2) = O\left(\sum_{i=1}^m N_i n_i\right) \text{ ops} \quad (14)$$

where the second term $O(n^2)$ has been dropped due to the strong dominance of the first term.

A.2. Phase 2 – The Phase 2 computation time is estimated based on three observations: First, each path is represented by a sequence of ρ_i postures. Second, each posture selection requires the evaluation of (13) for each of the a_i adjacencies of the most recently selected posture. Third, the evaluation of (13) involves collision probability values and arc-length evaluations. The collision probabilities are calculated off-line and are available without further calculations. The arc-length is similar to a Euclidean distance in an n_i -dimensional joint space, and thus has a computation time of $O(n_i)$ ops. Therefore, the total Phase 2 computation time is

$$T_2 = O\left(\sum_{i=1}^m \rho_i a_i n_i\right) \text{ ops} \quad (15)$$

A.3. Phase 3 – The computation involved in this phase is for the synthesis of the m chain paths into a single robot path. This synthesis process involves initialization of a work table, breadth-first search in the work table, and path extraction. The most costly of these is the breadth-first search. The work table is transformed into a graph, with vertices V representing all postures on the chain paths, $V = \prod_{i=1}^m \rho_i$, and edges E representing neighboring adjacency relationships, $E = 2mV$. Therefore, the computation time of this breadth-first search is

$$T_3 = O(E + V) = O\left(m \prod_{i=1}^m \rho_i\right) \text{ ops} \quad (16)$$

A.4. Total Computation Time – The total computation time using robot decomposition is $T_d = T_1 + T_2 + T_3$. Examining (14) and (15), we note that $\rho_i a_i$ represents the number of adjacencies of the ρ_i postures defining the chain ξ_i path. These $\rho_i a_i$ postures are a small subset of all N_i postures and thus $N_i \gg \rho_i a_i$ in all but pathological cases, implying that $T_1 \gg T_2$ and $T_d \approx T_1 + T_3$. Therefore,

$$T_d = O\left(\sum_{i=1}^m n_i N_i\right) + O\left(m \prod_{i=1}^m \rho_i\right) \text{ ops} \quad (17)$$

Now N_i and ρ_i can be compared by introducing

$$\rho = \left(\prod_{i=1}^m \rho_i \right)^{1/m} \quad (18)$$

which is the geometric mean of the number of postures on the path. We further note that a chain path consisting of ρ_i postures form a one dimensional curve in the n_i dimensional chain space that contains N_i postures, and thus

$$N_i = O(\rho_i^{n_i}) = O(\rho^{n_i}). \quad (19)$$

Equation (17) can now be written as

$$T_d = O\left(\sum_{i=1}^m n_i \rho^{n_i}\right) + O(m \rho^m) = O(\nu \rho^\nu) \text{ ops} \quad (20)$$

where

$$\nu = \max_i(n_i, m) \quad (21)$$

The first term in (20) involving n_i is associated with T_1 , and the second term involving m is associated with T_3 . Since the decomposition style \mathcal{D} dictates n_i and m , it also determines which component of computation time (i.e. T_1 or T_3) is dominant. This is intuitive since the proposed method decomposes the single n -dimensional robot path planning problem into a number of smaller problems of dimensions n_i in Phases 1 and 2, and one problem of dimension m in Phase 3. Extensive simulation studies using several different manipulators with results reported in Section V, support the above finding that T_2 is less than T_1 and almost always less than T_3 , and that the comparisons between T_1 and T_3 depends on the decomposition style \mathcal{D} .

Finally, explicit equations as opposed to the above order relationships, have been derived for the computation times of the three phases. These rather lengthy derivations are provided in [13], and the final results are given in Appendix B, which also confirm the above order relationship.

A.5. Benefit of Decomposition – In order to evaluate the effectiveness of decomposition, it is useful to compare planning time with and without decomposition. When the robot is not decomposed, the previously introduced phases 2 and 3 are replaced by a single search which generates a sequence of clear consecutively adjacent robot configurations joining the discretized start and goal. This single search can be either a local or a global search. Since a global search is much more computationally intensive than a local one, only the analysis of a local search with an undecomposed robot will be presented.

Since the mechanism of Phase 1 is similar with or without decomposition, from (14) the computation time of Phase 1 for the undecomposed robot is $O(N_u n)$ ops, where N_u is the number of discretized configurations, and n is the DOF of the robot. The local search selects a sequence of ρ_u collision free configurations defining the robot path. Its mechanism is identical to the Phase 2 search described in Section III-B. Thus, from (15) the computation time of the local search for the undecomposed robot is $O(\rho_u a_u n)$ ops,

where the subscript u stands for the undecomposed case. Once again, $\rho_u a_u$ represents the number of adjacencies of all configurations on the path, which are a subset of all N_u discretized configurations, thus $\rho_u a_u \ll N_u$, which leads to the total undecomposed computation time of

$$T_u = O(N_u n) \text{ ops} \quad (22)$$

Let us now compare the computation times of on-line planning with and without the use of decomposition. The N_u discretized configurations represent the n -dimensional C-space. Using the same argument as that for (19), $N_u = O(\rho^n)$. Substituting this into (22), we get

$$T_u = O(n \rho^n) \text{ ops} \quad (23)$$

From above, the computation time without decomposition is exponential in n which is the expected result, whereas from (20)-(21), the computation time with decomposition is exponential in $\nu = \max(n_i, m)$. Furthermore from close examination of Section II-A, $\nu \leq n - 2$ and is often $\frac{n}{2}$. So

$$T_d \ll T_u \quad (24)$$

which demonstrates the effectiveness of decomposition in reducing computation time. Note that this conclusion is reached despite using a semi-global search for the decomposed case and a local search for the undecomposed case.

Finally, the result in (20)-(21) is particularly important since it implies, for example, that path planning for an 8 DOF robot with the decomposition style $\mathcal{D} = \{323\}$ has the same order of computation complexity as that of an undecomposed 3 DOF.

B. Computation Space

If a fast algorithm requires more memory or *computation space* than is available in standard computers, then such an algorithm would not be practical. For this reason, this section analyzes the computation space of the proposed algorithm, and shows it to be superior due to robot decomposition. Since a robot path is a 1-dimensional curve in the n -dimensional C-space, the computation space required for the on-line planning is negligible compared to that for off-line modeling, and is therefore not examined here.

The computation space for the modeling consists of allocations for the discretized postures and the collision table. Each posture stores n_i joint variables, $a_i + 2$ addresses for the a_i adjacencies and their associated overhead, $\beta_i = n_i + 1$ separations to reference postures used for separation exclusions as described in Appendix A, and one collision probability γ_i . Therefore, the computation space $M_{P_{i,j}}$ for each posture is

$$M_{P_{i,j}} = n_i + (a_i + 2) + (n_i + 1) + 1 = a_i + 2n_i + 4 \text{ words} \quad (25)$$

where a *word* denotes the storage for either an integer or a floating point number. Note that in the worst case, there are many adjacencies for each discretized posture, i.e. $a_i \gg n_i$ in which case (25) reduces to $M_{P_{i,j}} = O(a_i)$ words. The

total computation space for the N_i discretized postures of all m chains is

$$M_{Pos} = O\left(\sum_{i=1}^m M_{P_{i,j}} N_i\right) = O\left(\sum_{i=1}^m a_i N_i\right) \text{ words} \quad (26)$$

Combining (19) and (26), we obtain

$$M_{Pos} = O\left(\sum_{i=1}^m a_i \rho^{n_i}\right) = O(\bar{a}_i \rho^{\bar{n}_i}) \quad (27)$$

where \bar{n}_i and \bar{a}_i are respectively the DOF and the number of adjacencies for the postures of the chain yielding the largest product $a_i \rho^{n_i}$.

The computation space associated with storing the binary collision state of a discretized configuration in a collision table cell is 1 bit. The space for storing the entire collision table is

$$M_{Tbl} = \prod_{i=1}^m N_i = N \text{ bits} \quad (28)$$

The total computation space required for the decomposed case is $M_d = M_{Pos} + M_{Tbl}$. In general since the collision table represents the n -dimensional C-space, M_{Tbl} is fairly comparable across all decomposition styles, and even without decomposition. However, the storage for postures M_{Pos} , depends on the decomposition style \mathcal{D} .

Similar to the derivation for $M_{P_{i,j}}$ above, in the undecomposed case each discretized configuration requires $M_{C_k} = O(a_u)$ words, where a_u is the average number of adjacencies for a configuration. The computation space for storing all configurations in the undecomposed case becomes

$$M_C = O(a_u N_u) = O(a_u \rho^n) \text{ words} \quad (29)$$

The collision states require $M_{Tbl} = O(N_u)$ bits, which is comparable to that with decomposition.

Let us next consider the benefit of decomposition in computation space requirements. It is seen from (27) and (29) that the computation space for the postures is exponential in \bar{n}_i for the decomposed case, and exponential in n for the undecomposed robot. Since from before $\bar{n}_i \leq n - 2$, and often $\frac{n}{2}$, this exponential growth is a significant difference, and thus

$$M_d \ll M_u \quad (30)$$

which shows the benefit of decomposition in the computation space savings. In the next section, we provide data in support of the above conclusion.

C. Success Rate and Design Parameters

In addition to computation time and space, the planning success rate is also important. The success rate depends on (a) the probability that the discretized postures can represent an existing path, and (b) the probability that the proposed planner can construct this path. For a given robot and environment, the success rate as well as computation time and space are dependent on the modeling

parameters, namely the decomposition style \mathcal{D} , the generation coefficient K_g , the acceptance coefficient K_a and the resolution R . The coefficients K_g and K_a allow for adjusting the posture generation mechanism which influence the density and distribution of the discretized postures at a given resolution R . While densely distributed postures improve the success rate, they produce more data for processing which in turn results in higher computation time and space.

The above considerations call for a set of modeling parameters $(\mathcal{D}, K_g, K_a, R)$ which minimize planning time T , while maximizing success rate σ . Optimization procedures are presented in [12] and [14] for this purpose. The optimization procedures involve several off-line preprocessing of different $(\mathcal{D}, K_g, K_a, R)$, each followed by solving a benchmark set of path planning problems. This optimization which requires multiple off-line preprocessing is very time consuming. However, the following reasoning narrows the choices of modeling parameters, and allows for reasonable estimates which are useful and perhaps near optimal.

The parameter K_g dictates the number and placement of candidate postures while K_a sets the minimum separation between accepted postures. From the definition of coverage ϕ , a well distributed set of postures is one in which any arbitrary posture is within a separation of $\frac{S_r}{2}$ from a discretized posture, and thus $K_g < 0.5$. On the other hand, (10) indicates that the higher the value of K_g , the fewer the number of candidate postures, leading to smaller computation time and space. These two conditions suggest using $K_g \approx 0.4$ to 0.5 . Extensive simulations have shown that there is generally a sharp drop in the number of discretized postures as K_g falls below K_a . This implies that in order to have sufficient discretized postures, one must have $K_g > K_a$, and therefore a selection rule for K_a is $K_a \approx K_g - 0.1$.

The decomposition style \mathcal{D} is mostly dependent on robot kinematics rather than on the environment. For most practical robots, the major joints, namely those with long links, do most of the work for collision avoidance. From the discussion on retrograde motion in Section III-D, successful maneuvering for collision avoidance requires cooperative motion of these joints. Depending on their number and kinematic arrangement, major joints are usually placed in the same chain, or in sparse environments divided up between two chains. This consideration together with the permissibility requirements of Section II-A, readily determines an appropriate decomposition style.

Finally, the user can usually specify the desired resolution based on the size of the arm and the spacing between obstacles. Generally, the resolution R is selected such that S_r in (6) is about $\frac{1}{2}$ to $\frac{1}{4}$ of the clearance between the arm and the obstacles in the most cluttered regions. For moderately cluttered environment, a resolution of $R \approx 0.1$ provides satisfactory results. The above guidelines lead to a quick selection of the modeling parameters without the use of an in-depth optimization procedure.

The sensitivity of planning time T and success rate σ to modeling parameters has been investigated based on both

analysis and simulations [14]. For small perturbations, T is very sensitive to both \mathcal{D} and R , while fairly insensitive at optimal values to K_g and K_a . On the other hand, σ is very sensitive to K_g , K_a , and R , and somewhat insensitive to \mathcal{D} .

V. SIMULATION RESULTS

The proposed path planning method was applied to two simulated manipulators, namely PUMA 560 and SRMS. Table I tabulates the Denavit-Hartenberg parameters [6], the ranges of motion of the joints, and the link lengths for the two simulated manipulators.

The 6 DOF Unimation PUMA 560 is the first of the two modeled robots. Joints 5 and 6 of PUMA 560 represent a compound joint, which result in the length of link 5 being zero. In order for joints 4 and 5 to have a more pronounced effect on the arm collision with the environment, the PUMA is modeled with a cylindrical extension on its end-effector to represent a tool. This extension is 24.75 cm long, and 52 mm in diameter.

The Shuttle Remote Manipulation System (SRMS) is the 6 DOF manipulator used on board the Space Shuttle cargo bay. The first link of the SRMS is a short link which rotates about its longitudinal axis. The next two links are extremely long and thin (aspect ratios of up to 18.5), and represent its upper arm and forearm. The last three links form the SRMS's wrist. Due to the extreme torques required to move the long links of the SRMS, the links cannot operate outside the weightless environment of space.

In order to examine the behavior of the robots, the environment shown in Fig. 4 is employed. It consists of six prismatic objects arranged in a vertical pattern surrounding the origin, and sandwiched between two vertical walls. The size of space between these walls is $3556 \times 1334 \times 2667$ cm. This space contains four thick prisms which fit into $667 \times 1334 \times 667$ cm, and two thin prisms fitting into $667 \times 1334 \times 178$ cm. The base of the arm is located at the center of the environment. Considering the length of the fully extended SRMS which is $l_1 = 1532$ cm, the environment is tight and cluttered making path planning a challenging task. Note that the shrouds of the SRMS have been expanded by $\frac{S_c}{2}$ from each link surface as discussed before. The same prisms environment is used for PUMA, but is scaled down by a factor of approximately 13 to make the environment equally challenging for the PUMA which is much smaller than the SRMS, and has a fully extended arm length of $l_1 = 115$ cm.

Let us first compare the computation space needed to store various information such as the discretized postures and the collision table at different decomposition styles. Table II compares the number of discretized postures, computation space for the SRMS manipulator for generation and acceptance coefficients of $K_g = 0.45$, $K_a = 0.4$, two resolutions and four decomposition styles. The undecomposed case $\mathcal{D} = \{6\}$ also presented, requires significantly higher computation space than any of the decomposition styles $\mathcal{D} = \{24\}, \{33\}, \{222\}, \{42\}$. This verifies the significant benefit of decomposition, and the impracticality of

TABLE I
GEOMETRICAL SUMMARY

	i	α_{i-1} (deg)	a_{i-1} (cm)	d_i (cm)	θ_i range (deg)	length (cm)
PUMA	1	0	0.0	0.0	320	68.6
	2	-90	0.0	0.0	266	73.6
	3	0	41.0	13.65	284	47.9
	4	-90	0.0	46.8	280	14.3
	5	90	0.0	0.0	200	0
	6	-90	0.0	0.0	532	24.8
SRMS	1	0	0.0	0.0	350.8	30.5
	2	-90	0.0	0.0	137.8	637.8
	3	0	637.8	0.0	153.2	706.1
	4	0	706.1	0.0	228.8	45.7
	5	90	45.7	0.0	229.2	76.2
	6	-90	0.0	0.0	880.0	66.0

using the undecomposed model for the given resolutions. Note that basal chains (closest to the base) produce substantially more postures compared to distal chains since from (10) small changes in basal chain joints produce larger work space motions. The PUMA manipulator also shows similar benefits of decomposition.

TABLE II
COMPUTATION SPACE FOR DIFFERENT DECOMPOSITION STYLES AND RESOLUTIONS FOR SRMS

R	\mathcal{D}	N_i	$M(MBytes)$
0.16	{6}	{683942}	1212.030
	{222}	{1510, 66, 23}	0.414
	{24}	{1510, 736}	0.359
	{33}	{9676, 77}	1.720
	{42}	{42710, 23}	15.875
0.10	{6}	{6742385}	14213.811
	{222}	{3896, 1781, 32}	3.107
	{24}	{3896, 3152}	2.386
	{33}	{38384, 193}	7.609
	{42}	{297199, 32}	129.330

We now show two typical paths constructed for the SRMS in the above prisms environment with expanded link shrouds. Fig. 4 shows the first scenario which starts with the “elbow”, namely the joint between its two longest links, pointing to the left. The motion ends with the elbow pointing to the right with the wrist up and rotated. Also shown is an intermediate configuration which shows the elbow rotated toward the front wall and dipped down to avoid the front wall. Using $\mathcal{D} = \{24\}$ and $R = 0.1$, this path was constructed in 0.019 second of CPU time on a Pentium II PC running at 333 MHz.

Fig. 5 shows a zoomed view of the SRMS in the prisms environment, this time starting fully extended and pointing to the left. The motion ends with the elbow flexed and lifted up, and the wrist rotated. Also shown are two inter-

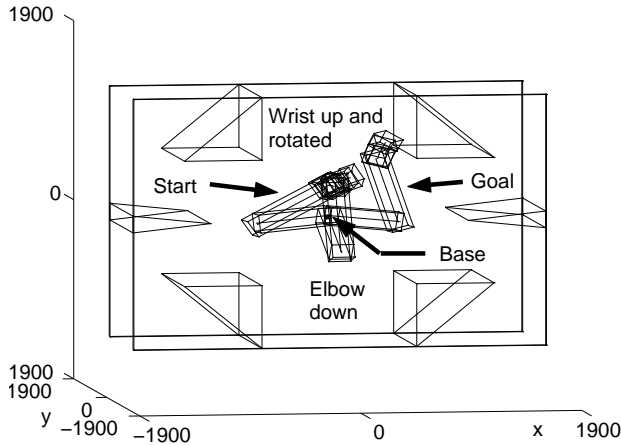


Fig. 4. SRMS avoiding front wall and prisms (dimensions in centimeters).

mediate configurations depicting the arm being retracted to avoid the prisms. This is an example of the retrograde motion described in Section III-D. In this case, successful planning requires coordinated motion of joints 2 and 3 which actuate the two longest links. Therefore, joints 2 and 3 must be in the same chain, and thus the decomposition style $\mathcal{D} = \{33\}$ was used to plan this path. The entire path was planned in 0.026 seconds of CPU time on a Pentium II PC running at 333 MHz.

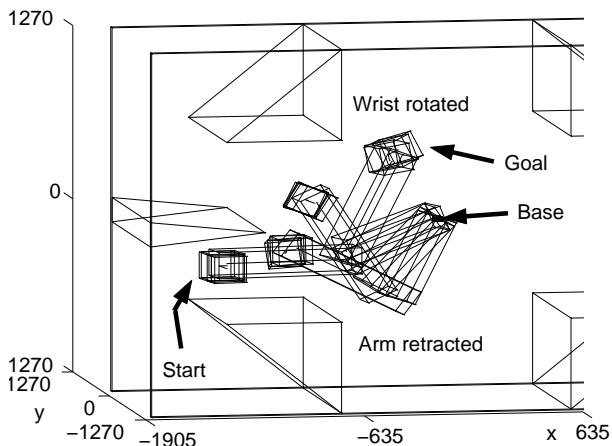


Fig. 5. SRMS extracting arm out of prisms and rotating up (dimensions are in centimeters).

We now compare the computation times of the three phases of the planner at different decomposition styles and resolutions, each averaged for path planning attempts for a set of 250 randomly chosen start and goal configurations. Table III, shows average computation times of the three phases T_1 , T_2 , and T_3 , for the SRMS manipulator at the four different decomposition styles \mathcal{D} , and at two resolutions. This table supports the discussion in Section IV-A.4 by showing that Phase 2 computation time $T_2 < T_1$, and that the comparison between T_1 and T_3 depends on the

decomposition style.

TABLE III
COMPUTATION TIMES FOR 3 PHASES OF THE SRMS MANIPULATOR AT 4 DECOMPOSITION STYLES AND 2 RESOLUTIONS.

Decomp. Style \mathcal{D}	{222}		{24}	
	Resolution R			
	0.160	0.100	0.160	0.100
T_1 (sec)	0.0034	0.0048	0.0055	0.0102
T_2 (sec)	0.0003	0.0005	0.0006	0.0017
T_3 (sec)	0.0066	0.0146	0.0026	0.0078
T (sec)	0.0103	0.0199	0.0086	0.0197
Decomp. Style \mathcal{D}	{33}		{42}	
	Resolution R			
	0.160	0.100	0.160	0.100
T_1 (sec)	0.0093	0.0303	0.0266	0.1510
T_2 (sec)	0.0005	0.0008	0.0012	0.0022
T_3 (sec)	0.0006	0.0011	0.0003	0.0009
T (sec)	0.0104	0.0322	0.0282	0.1541

Finally, Table IV provides average planning time and success at $\mathcal{D} = \{24\}$ and $R = 0.100$ for the PUMA and the SRMS. The first two rows refer to the above prisms environment, and the third row is for the shuttle cargo bay environment to be discussed shortly. For each of these robot and environment pairs, path planning is attempted for a set of 250 randomly chosen start and goal configurations. Note that the lower success rate of the SRMS in the prisms environment is partly due to inability of the planner in finding existing collision free paths, but is also due to physically non-existent free paths for some of the randomly selected start and goal configurations. Tables III and IV show that very short planning times can be obtained using the proposed method. Although the number ρ_d of discretized configurations on the path, is dependent on the environment, the start and goal pair, and the modeling parameters, it is worthwhile mentioning that typical ρ_d for all cases in Table IV range between 20 and 50 configurations. Finally, we have visually inspected the simulated arm motion for many of the planned paths and found the motion to be smooth and elegant.

TABLE IV
COMPUTATION TIMES AND SUCCESS RATES

Robot	Envi- ronment	Time (sec)		Success Rate
		Average	Std Dev	
PUMA	Prisms	0.0120	0.0220	250/250
SRMS	Prisms	0.0197	0.0390	208/250
SRMS	Cargo	0.0414	0.0740	237/250

We now use the SRMS in its mock-up cargo bay environment as shown in Fig. 6. The cargo bay environment consists of a payload bay of dimension $2477 \times 762 \times 572$ cm, a rectangular airlock with dimensions $381 \times 381 \times 381$ cm above the forward region of the payload bay, a rectangular free flyer payload of dimensions $381 \times 305 \times 191$ cm hovering over the center of the payload bay, and a rectan-

guloïd berthed payload at the bottom of the payload bay at its middle section. The base of the arm is located 572 cm above the payload bay floor, 114 cm off the centerline of the payload bay on the port (left) side and 76 cm behind the back wall of the cabin (cockpit) region. This cargo bay environment is typical where the SRMS would be employed. For this environment, we used the same design parameters as above, i.e. $\mathcal{D} = \{24\}$ and $\{33\}$, $K_g = 0.45$, $K_a = 0.40$, and $R = 0.1$.

Figures 6 and 7 depict two views of a typical path constructed in the cargo bay environment. The expanded arm shroud is not shown to enable a better view of the environment. First, the SRMS starts from the starboard (right side) on the outside of the payload bay. It lifts up to clear the starboard payload bay wall, then approaches the free flyer payload hovering over the top of the payload bay. It then stops its lateral motion to lift its forearm (distal boom) over the top of the free flyer while keeping its upper arm (basal boom) under the airlock which is directly over the base of the SRMS. Once the forearm is high enough to clear the free flyer, it begins to move laterally again until it reaches the goal configuration. Note that the planning also involves motion of the wrist between start and goal configurations. The motion is very smooth, partly due to path conditioning mentioned in Section III-D.1.

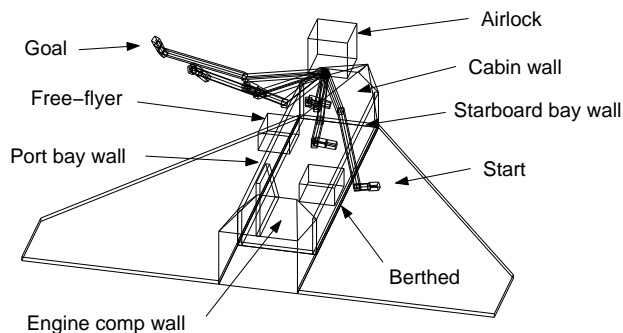


Fig. 6. Sequence of SRMS configurations from start to goal in the Shuttle cargo bay

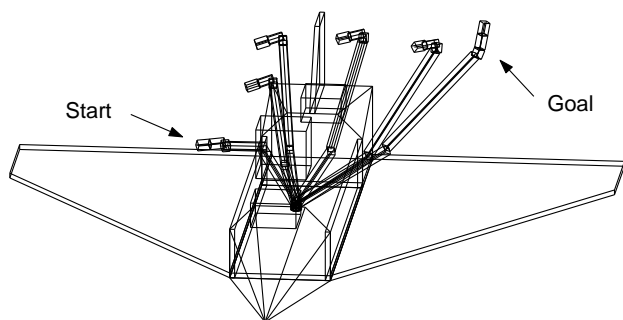


Fig. 7. Front view of the SRMS in Shuttle cargo bay

Some 250 path planning trials were performed in the cargo bay environment with randomly chosen start and goal configurations. The planning times and success rates

are shown in Table IV for these trials, and show similarly short planning times and good success rates. Note that some of the failures are due to non-existent paths, rather than the planning method.

VI. DISCUSSION AND CONCLUSIONS

An efficient path planning method has been presented for manipulators with 3-D links operating in cluttered 3-D environments. In this work, exponential growth of the planning time with robot DOF is circumvented by decomposing a high DOF robot planning problem into a number of low DOF chain planning problems. The chain paths are then synthesized to obtain the robot path. As a result of this particular decomposition, the n -dimensional search is essentially reduced to m low dimensional local searches and one m -dimensional global search where m is the number of chains. The planning time is, therefore, significantly reduced because it becomes exponential in chain DOF which is a fraction of the robot DOF. Typical planning times of fraction of a second on a PC or workstation is obtained for typical 6 DOF manipulators operating in relatively cluttered environments. Since the robot is decomposed, and the planner employs a local search in Phase 2, it is not complete in the sense that there may exist cases where an existing path eludes the planner. In order to substantially reduce such cases, the strategies of path conditioning and backtracking are presented, and the concept of retrograde motion is introduced.

Let us now compare the planner presented here with other strategies. The proposed method has a similarity to the approximate cell decomposition method, since the C-space is discretized, with each configuration having adjacencies and a collision state. In typical cell decompositions, a regular grid is utilized uniformly discretizing the C-space. However, distances map heterogeneously from the C-space to the work space where the actual robot motion takes place and where collisions must be avoided. As a result, certain regions in the work space would be under-represented by the discretized configurations whereas other work space regions may contain too many unnecessary configurations. In the proposed method, the posture generation and acceptance procedure ensures good coverage and connectivity without undue numbers of discretized postures, thus reducing the planning time and maximizing the success rate.

Phase 2 of the planner employs a local search and constructs each chain path by locally optimizing an objective function, and is therefore similar to the potential field method. However, unlike the potential field method that constructs the robot path based purely on a local search, the proposed planner is semi-global since it employs low dimensional local searches in Phase 2 for the chains followed by a global search to synthesize the chain paths in Phase 3. It is, therefore, less prone to getting trapped in local minima.

Now consider the strength of the proposed planner over randomized and probabilistic planners. The randomized planner in [1] develops a potential function in the C-space which is incrementally searched for the path. This poten-

tial field function may have local minima in which case a random walk is initiated to escape. The randomized method requires a high dimensional search which together with random walk can cause high computation times. This is in contrast to the proposed planner that requires only low dimensional searches.

The probabilistic road map method [17]-[18] generates a road map from a random set of collision free configurations in C-space. Path planning then consists of connecting the start and goal configurations to the road map, followed by connecting the road map entries and exits. These connections require on-line collision checking, which are computationally intensive. In contrast, all collision checking in the proposed planner is done off-line, and the information is stored for fast look up during the on-line planning. In addition, when probabilistic road map runs into difficulties, it begins random walks which generally require more steps than the proposed method. On the other hand, the drawback of the current implementation of our planner is that it requires a priori knowledge of the environment. In most practical applications, only a fraction of the environment may change and the remainder consists of fixed objects. It is possible to develop another program that would run in parallel to update the collision table online based on sensory data or other information. Fast on-line planning of the proposed method provides good margin to run other processes in parallel, such as collision table updating.

The sequential search in [9] and [10] decomposes the robot planning problem into a sequence of several link planning problems. The path of each link is based on the path of all previous links, through the construction of parametric spaces which require on-line collision checking thus making planning very computationally intensive. This is unlike the proposed planner which performs all collision checking off-line, while the on-line planning of each chain is done independently of the other chains, thus having the potential for parallel implementation.

Let us now compare the computation time of the proposed method with the existing methods. The probabilistic road map [17]-[18] implemented on a DEC Alpha machine requires 5 seconds or more for 4 and 5 DOF planar polygonal manipulators, and 20 to 80 seconds for a 7 DOF stick robot, both in 2-dimensional work spaces. SANDROS of Hwang and Chen [5], [15] implemented on a 200 MHz SGI computer requires 14 seconds for a 4 DOF robot, and 22 seconds for the 5 DOF Adept I robot. The quoted time for the sequential search method [9] is 10 minutes for a 4 DOF planar manipulator, and as much as one hour for a 6 DOF planar robot. In contrast the proposed planner requires only a few hundredth of a second to plan a path for a 6 DOF non-planar robot in a relatively complex 3-dimensional work space.

We conclude the discussion by mentioning some of the limitations of the proposed planner. Since the planner derives its benefits from reducing the dimensionality of the problem by decomposition, it loses its utility for low (e.g. less than 4) DOF manipulators due to the overhead associated with decomposition. On the other hand, since the

generation of postures starts with a uniform division of the joint space, the memory limitation of a standard computer imposes a bound on the resolution of the planner for very high (e.g. more than 12) DOF manipulators. Finally, due to the loss of “big picture” perspective associated with decomposition and the need for sophisticated retrograde motions in highly cluttered environments, the proposed planner loses its effectiveness for manipulators with many long links in cluttered environments.

APPENDIX A – METRIC EXCLUSIONS

Separation is a computationally expensive metric to evaluate. The important triangle inequality property, which for metric $S(\cdot, \cdot)$ can be rewritten as

$$|S(a, b) - S(b, c)| \leq S(a, c) \quad \forall a, b, c \quad (31)$$

is used to derive an efficient exclusion algorithm, which eliminates many of its unnecessary evaluations. Consider the following problem which is solved during posture generation of off-line modeling, and Phase 1 of on-line planning. Suppose there are N points q_i , $i \in \{1, \dots, N\}$, in an n -dimensional normed linear space $\{q\}$, with metric $S(q_i, q_j)$ defined. Suppose further that for a given q_i , the values for all q_j satisfying

$$\{q_j : S(q_i, q_j) \leq S_{\max}\} \quad (32)$$

are sought. Alternatively, the values of $S(q_i, q_j)$ for those q_j further than S_{\max} from q_i are not needed, and their evaluations may be eliminated. In order to conservatively determine the disposition of (32) without initially evaluating $S(q_i, q_j)$, the following exclusion test has been created using the triangle inequality of (31).

Denote a point q_r as a *reference* point. Suppose the values $S(q_i, q_r)$ and $S(q_j, q_r)$ are already known, and differ by more than the preselected constant S_{\max} for some q_i and q_j . From (31), this is restated as

$$S_{\max} < |S(q_i, q_r) - S(q_r, q_j)| \leq S(q_i, q_j) \quad (33)$$

This asserts that $S_{\max} < S(q_i, q_j)$ and eliminates the necessity of evaluating $S(q_i, q_j)$ for all q_j such that

$$\{q_j : S_{\max} < |S(q_i, q_r) - S(q_r, q_j)|\} \quad (34)$$

On the other hand all q_j satisfying

$$\{q_j : |S(q_i, q_r) - S(q_r, q_j)| \leq S_{\max}\} \quad (35)$$

are not excluded, and their $S(q_i, q_j)$ must still be evaluated. The set of points q_j satisfying (35) form a thick closed shell centered at q_r in the $\{q\}$ space, as shown in Fig. 8. The dashed lines represent an isometric curve of all q_j satisfying $S(q_j, q_r) = S(q_i, q_r)$. The shape of this shell depends on the metric $S(\cdot, \cdot)$, and may be non-spherical and irregular unlike the illustration in Fig. 8. From (35), the thickness of this shell is $2S_{\max}$.

Employing a number β of randomly positioned reference points q_r is highly beneficial. Each additional q_r introduces

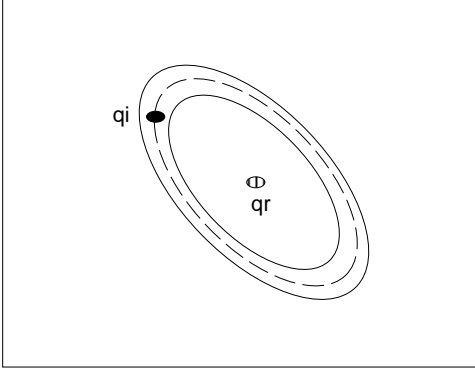


Fig. 8. The set of all q_j that are *not* excluded.

a new constraint, or alternatively a new closed shell, making the exclusion test more effective. The result is that only points q_j satisfying all β constraints, or those in the intersection of all β shells, must have their separations evaluated. However, each additional q_r also requires the additional computation time and space associated with evaluating and storing $S(q_r, q_k), \forall k$. Therefore β is chosen to enhance the effectiveness of the exclusion without adding significant overhead. In [14], the optimal number of reference points is shown to be $\beta = n + 1$, where n is the dimension of the space. This leads to the following exclusion test.

Suppose $\beta = (n + 1)$ reference points q_r are defined and evenly separated in an n -dimensional $\{q\}$ space. Suppose further that the values of $S(q_r, q_j)$ for all q_r and q_j are known. The exclusion test $E(\cdot, \cdot)$ is defined as

$$E(q_i, q_j) = \max_r |S(q_i, q_r) - S(q_r, q_j)| \quad (36)$$

where $r \in \{1, \dots, (n + 1)\}$, and from (33) satisfies

$$E(q_i, q_j) \leq S(q_i, q_j) \quad (37)$$

The above development outlines an exclusion test that yields a lower bound for separations, and is used to eliminate unnecessary evaluations. The number of separation evaluations in the posture generation process can be shown to be reduced from $O(N_i^2)$ to $O(N_i n_i)$, and for Phase 1 of on-line planning from $O(N_i)$ to $O(n_i)$, both for chain ξ_i , where N_i is the number of postures for chain ξ_i and n_i is the dimension of its joint subspace [14].

APPENDIX B – COMPUTATION TIME EQUATIONS

In this appendix we provide the computation time equations for the three phases of the path planner. Detailed derivations can be found in [13].

Phase 1 consists of (a) searching through postures of the chains, and evaluating the exclusion tests to find likely candidates for the discretized postures nearby the actual start and goal, and (b) combining them to form discretized configurations C_s and C_g , and verifying that their collision states and their separations to the actual start \tilde{C}_s and goal

\tilde{C}_g are as desired. The maximum Phase 1 computation time is obtained as [13]

$$T_1 = 4 \sum_{i=1}^m N_i n_i t_{add} + 2(n + m + \delta) \times \\ [(482n + m) t_{add} + 210n t_{mult} + m t_{div} + 8n t_{sin}]$$

where t_{add} , t_{mult} , t_{div} and t_{sin} are the computation times associated with an addition, a multiplication, a division, and a trigonometric function evaluation, respectively. The term δ is typically smaller than $(n + m)$, and is a count of the separation evaluations used for verifying a discretized configuration representing an actual start or goal. The fastest growing term in T_1 is the $\sum_{i=1}^m N_i n_i$ term, which confirms (14). Note that without the use of exclusion tests of Appendix A, the $(n + m + \delta)$ factor would become $(\sum_{i=1}^m N_i + \delta)$ which significantly increases T_1 .

Phase 2 computation consists of inspecting all adjacencies to find their arc-lengths to the goal, retrieving collision probability values, and determining the objective function (13). It is found that [13]

$$T_2 = \sum_{i=1}^m (\rho_i - 1) [(5n_i + 2)a_i + 1] t_{add} \\ + (2n_i a_i + 1) t_{mult} + a_i t_{div} + a_i t_{sqrt}$$

where t_{sqrt} is the time needed for square root computation. Note that that dominant term in T_2 is $\sum_{i=1}^m \rho_i a_i n_i$, which is consistent with (15).

Finally, Phase 3 computation consists of initializing the work table, performing the breadth search, and extracting the robot path. It is found that [13]

$$T_3 = [(6m + 1)\rho^m + 8m^2 + m + 4\rho_d] t_{add} + 2m^2 t_{mult}$$

where ρ_d is the number of configurations on the robot path. The dominant term in T_3 is $(m\rho^m)$ which is consistent with (16), and (18).

Note that the theoretical computation times given above are shorter than the actual times, e.g. those reported in Tables III and IV, due to software implementation overhead. However, they show the terms involved and their relative contributions to the on-line planning time.

REFERENCES

- [1] J. Barraquand, J.-C. Latombe, *Robot Motion Planning: A Distributed Representation Approach*, International Journal of Robotics Research, Vol. 10, No. 6, December 1991, pp. 628–649.
- [2] J. Barraquand, L. Kavraki, J.-C. Latombe, R. Motwani, T.-Y. Li, and P. Raghavan, *A Random Sampling Scheme for Path Planning*, The International Journal of Robotics Research, Vol. 16, No. 6, December 1997, pp. 759–774.
- [3] R. A. Brooks and T. Lozano-Perez, *A Subdivision Algorithm in Configuration Space for Findpath With Rotation*, Proceedings of International Joint Conference on Artificial Intelligence, Germany 1983, pp. 799–806.
- [4] J. F. Canny, *The Complexity of Robot Motion Planning*, MIT Press, 1988.
- [5] P. C. Chen, Y. K. Hwang, *SANDROS: A Dynamic Graph Search Algorithm for Motion Planning*, IEEE Transactions on Robotics and Automation, Vol. 14, No. 3, June 1998, pp. 390–403.
- [6] J. Denavit and R.S. Hartenberg, *A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices*, Journal of Applied Mechanics, June 1955, pp. 215–221.

- [7] E. Gilbert, D. W. Johnson, and S. Sathiy Keerthi, *A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space*, IEEE Journal of Robotics and Automation, Vol. 4, No. 2, April 1988, pp. 193–203.
- [8] K. K. Gupta, *Fast Collision Avoidance for Manipulator Arms: A sequential Search Strategy*, IEEE Transactions on Robotics and Automation, Vol. 6, No. 5, October 1990, pp. 522–532.
- [9] K. K. Gupta and Z. Guo, *Motion Planning for Many Degrees of Freedom: Sequential Search with Backtracking*, IEEE Transactions on Robotics and Automation, Vol. 11, No. 6, December 1995, pp. 897–906.
- [10] K. K. Gupta and X. Zhu, “Practical global motion planning for many degrees of freedom – A novel approach with sequential search framework,” J. Robotics Systems, vol. 12, no. 2, 1995, pp. 105–117.
- [11] A. Hourtash and M. Tarokh, *Optimal Robot Decomposition for Fast Path Planning*, Proceedings IEEE International Conference on Advanced Robotics, Monterey, CA, 1997, pp. 339–345.
- [12] A. Hourtash and M. Tarokh, *Efficient Path Planning by Optimal Configuration Generation Using Evolutionary Programming*, Proceedings World Automation Conference, Anchorage, Alaska, 1998.
- [13] A. Hourtash and M. Tarokh, *Computation Time Analysis of Path Planning by Decomposition*, Technical Note No. 5-2001, Intelligent Machines and Systems Laboratory, Department of Computer Science, San Diego, State University, August 2001.
- [14] A. Hourtash, *Robot Path Planning by Decomposition*, Ph.D. dissertation, University of California, San Diego, CA, 1999.
- [15] Y. K. Hwang and Pang C. Chen, *A Heuristic and Complete Planner for the Classical Mover’s Problem*, Proceedings IEEE International Conference on Robotics and Automation, 1995, pp. 729–736.
- [16] L. E. Kavraki, *Computation of Configuration-Space Obstacles Using the Fast Fourier Transform*, IEEE Transactions on Robotics and Automation, Vol. 11, No. 3, June 1995, pp. 408–413.
- [17] L. E. Kavraki, *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*, IEEE Transactions on Robotics and Automation, Vol. 12, No. 4, August 1996, pp. 566–580.
- [18] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, *Analysis of Probabilistic Roadmaps for Path Planning*, IEEE Transactions on Robotics and Automation, Vol. 14, No. 1, February 1998, pp. 166–171.
- [19] O. Khatib, *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, International Journal of Robotics Research, Vol. 5, No. 1, Spring 1986, pp. 90–98.
- [20] D. E. Koditschek, *Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations*, Proceedings IEEE International Conference on Robotics and Automation, 1987, pp. 1–6.
- [21] J.-C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, 1991.
- [22] M. C. Lin and J. F. Canny, *A Fast Algorithm for Incremental Distance Calculation*, Proceedings IEEE International Conference on Robotics and Automation, 1991, pp. 1008–1014.
- [23] T. Lozano-Perez, *Spatial Planning: A Configuration Space Approach*, IEEE Transactions on Computers, Vol. C-32, No. 2, February 1983, pp. 108–120.
- [24] T. Lozano-Perez, *A Simple Motion-Planning Algorithm for General Robot Manipulators*, IEEE Journal of Robotics and Automation, Vol. RA-3, No. 3, June 1987, pp. 224–237.
- [25] A. A. Maciejewski, *Path Planning and the Topology of Configuration Space*, IEEE Transactions on Robotics and Automation, Vol. 9, No. 4, August 1993, pp. 444–456.
- [26] S. Quinlan, *Efficient Distance Computation between Non-Convex Objects*, Proceedings IEEE International Conference on Robotics and Automation, San Diego, CA, 1994, pp. 3324–3329.
- [27] E. Rimon and D. Koditschek, *Exact Robot Navigation Using Artificial Potential Functions*, IEEE Transactions on Robotics and Automation, Vol. 8, No. 5, October 1992, pp. 501–518.
- [28] M. Tarokh, *Fast Path Planning for Robot Manipulators*, Proceedings IEEE/JRS International Conference on Intelligent Robots and Systems, Raleigh, NC, July 1992, pp. 663–668.
- [29] M. Tarokh, *Fast Path Planning for Robot Manipulators by Formation-Posture Decomposition*, Proceedings IEEE/JRS International Conference on Intelligent Robots and Systems, Pittsburgh, PA, August 1995, Vol. 2, pp. 138–143.
- [30] M. Tarokh, *Implementation of a Fast Path Planner on an Industrial Manipulator*, Proceedings IEEE International Conference on Robotics and Automation, Minneapolis, MN, April 1996, pp. 436–441.
- [31] M. Tarokh and A. Hourtash, *Path Planning by Robot Decomposition and Parallel Search*, Proceedings IEEE International Conference on Robotics and Automation, Albuquerque, NM, 1997, pp. 562–568.
- [32] D. Zhu and J.-C. Latombe, *New Heuristic Algorithms for Efficient Hierarchical Path Planning*, IEEE Transactions on Robotics and Automation, Vol. 7, No. 1, February 1991, pp. 9–19.

Arjang Hourtash was born in Tehran, Iran in 1969. He received the B.S. and M.S. degrees in Nuclear and Mechanical Engineering from University of California at Santa Barbara in 1989 and 1991, and Ph.D. in Engineering Sciences, emphasis in robotics, from University of California at San Diego in 1999. During 1994-2000 he served on the technical staff of Hewlett-Packard and two other high tech companies in San Diego. He is the founder, and currently the president of Simplify Robotics Inc., in Houston, Texas, and consults at the Engineering Robotics Directorate of NASA Johnson Space Center.

Mahmoud Tarokh received the B.S., M.S. and Ph.D. degrees in electrical and computer engineering, respectively, from Tehran Polytechnic University, Iran, University of Birmingham, England, and University of New Mexico. He has published extensively in the area of robot path planning and control. His recent research interest and work has been in field of intelligent navigation and control of rovers and their applications to terrestrial and space explorations. Professor Tarokh has held academic positions at Sharif University, Iran, University of Colorado, Boulder, University of New Mexico and University of California, San Diego. He is currently is a professor of computer science and Director of Intelligent Machines and Systems Laboratory, at San Diego State University.