Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems Raleigh, NC July 7-10, 1992

Motion Planning for Redundant Branching Articulated Figures with Many Degrees of Freedom

Wallace S. Ching, Norman I. Badler
Computer Graphics Research Laboratory
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pennsylvania 19104-6389

ABSTRACT

A fast algorithm is presented that can handle the motion planning problem for articulated figures with branches and many degrees of freedom. The algorithm breaks down the degrees of freedom of the figure into *Cspace groups* and compute the free motion for each of these groups in a sequential fashion. It traverses the tree in a depth first order to compute the motion for all the branches. A special playback routine is then used to traverse the tree again in a reverse order to playback the final motion. The planner runs in linear time with respect to the total number of Cspace groups without backtracking. We believe that the planner would find a path in most cases and is fast enough for practical use in a wide range of applications.

1 Introduction

This paper presents an implementation of a new motion planning algorithm for general redundant branching articulated figures with many degrees of freedom. The algorithm has a number of advantages. For example, it is fast for articulated figures with many degrees of freedom (DOFs) and it can handle branches on the figures.

The motion planning problem is generally defined as finding a path from a specified starting robot configuration to a specified goal configuration that avoids collisions with a known set of stationary obstacles residing in the robot's workspace. Collision-free path planning has applications in a variety of fields such as robotics task planning, computer aided manufacturing, human figure modelling and ergonomics studies. The development of practical motion planning algorithms can reduce significantly the burden of the operator and the total programming time required. Our interest for motion planning on articulated branching figures arise from our general interest in human figure modelling and motion simulation. In this case, the upper body of the human figure will be anaylsed as a tree structure.

A great deal of research has been devoted to the motion planning problem within the last 10 years [12] [13] [5] [4] [6] [7] [11] [9]. However, many of these algorithms deal with manipulators with relatively few degrees of freedom such as the mobile robots which typically have three degrees of freedom and the PUMA type of robots which have six. Many of these algorithms are based on the use of the configuration space (C space) which is the space of the degrees of freedom of the robot. The inherent difficulties with this approach is due to the high dimensionality of the C space. It is well known that the worst case time bound for motion planning for a robot arm is exponential in the dimensionality of its C space [15] [16]. It is only during the last few years that motion planning algorithms that can handle manipulators with many degrees of freedom have been presented [2] [1] [3] [10] [8].

However, very few of the work consider articulated figures with branches. Barraquand *et al* gave an example involving a manipulator with 2 branches which is not handled in an explicit and general manner [2][1][3]. A gain in efficiency is obtained as a result of the clever selection of potential functions and heuristics. However, it is not clear how these can be selected in general.

Faverjon *et al* [8] presented a method which partitions the free space into octrees and uses some probability measures to cut down the search tree during the A* search. Gupta [10] has presented another technique to handle sequential linkages with many degrees of freedom. It is a sequential search technique which basically treats the individual degrees of freedom one by one instead of considering all of them together. The initial stage of our path planner is based on his work. We will look at the details in later sections.

2 The Approach

The initial stage of our path planner is derived from the sequential search strategy presented by Gupta [10]. The idea is that since it is computationally intractable to deal with all the degrees of freedom at a time, it may be better to deal with only a subset of them at one time.

Gupta's algorithm plans the motion of each link successively, starting from the base link. When the motion of links till link i has been planned, the path of one end (the proximal end) of link $i+1$ will then be determined. The motion of link $i+1$ is then planned along this path by controlling the degree of freedom associated with

[0]This research is partially supported by Lockheed Engineering and Management Services (NASA Johnson Space Center), MOCO Inc., NSF CISE Grant CDA88-22719, and ARO Grant DAAL03-89-C-0031 including participation by the U.S. Army Human Engineering Laboratory, Natick Laboratory, TACOM, and NASA Ames Research Center.

0-7803-0737-2/92$03.00 1992©IEEE

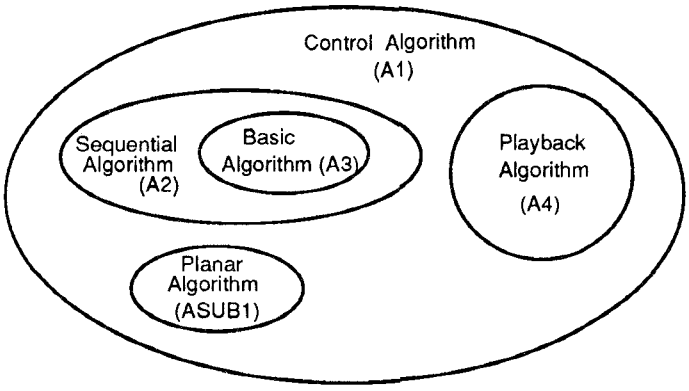

Figure 1: Different modules and algorithms in the path planning system

it, which is a two-dimensional motion planning problem. This strategy results in one 1-dimensional (the first link) and $(n-1)$ 2-dimensional planning problems instead of one n-dimensional problem for a n-link (i.e. n degrees of freedom) manipulator arm.

The domain of articulated figures we are interested in involves models of human beings which are characterized by large number of degrees of freedom and branching of the linkages into a tree structure. There are many unique problems encountered when dealing with these types of figures. Gupta's algorithm alone cannot resolve all these issues. We will explain the rest of our path planner in later sections and see how they address all of these special issues.

2.1 Overview

We will name the root of the tree structure the *base point*. Our algorithms will be demonstrated using human figure models. We will basically focus on the upper body of the figure. In this case, the base point will be at the *waist* joint of the figure.

Our system adopts a modular design in that it is made up of a number of modules each of which is based on an algorithm (Figure 1).

On a more global perspective, the path finding procedure can be viewed as consisting of two phases: the *computation* phase and the *playback* phase.

The overall path planning procedure is outlined as follows:

- **Computation** Phase:

  1. Partition the degrees of freedom of the articulated figure into *Cspace groups* (or *Cgroups*) (more details in the next section) according to a grouping scheme.
  2. Impose an ordering of the Cgroups. This will also be the order in which we will traverse the tree structure and compute the motion for the joints.
  3. Invoke the *control* algorithm that handles traversal of the tree and finding the final collision free path. This algorithm will actually call upon a subsidiary algorithm, the *sequential algorithm*, to compute the free path along each branch of the tree structure. The *sequential* algorithm will in turn call another subsidiary algorithm, the *basic algorithm*, to compute the path for the DOFs within each Cgroup.

     The details of these algorithms will be covered in the next few sections.

- **Playback** Phase:

  After all the Cspace groups have been considered, the control algorithm will call upon a special playback routine which will traverse the tree structure in a reverse order, collect and coordinate all the information about the free paths computed for each group and finally playback the overall collision free path for the whole figure in discrete time frames. The routine is based on the *playback* algorithm (Fig. 1).

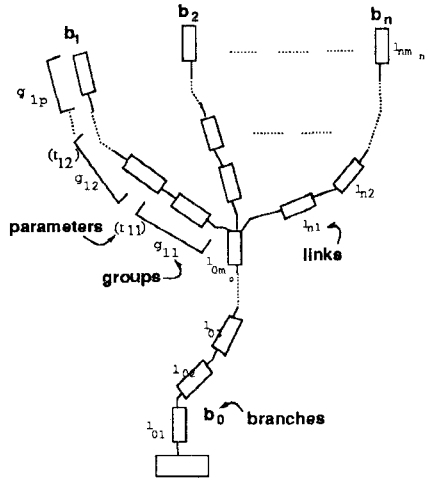

Figure 2: The redundant branching articulated figure considered in this work

The translational movement of the articulated figure as a whole on a plane can also be planned with our planner. In this case, the figure is simply treated as a mobile robot with two degrees of translational freedom and one degree of rotational freedom.

The main algorithm focuses on a subset of the tree structure which is shown in Fig. 2. The whole tree structure is then handled by recursively applying the main algorithm.

Fig. 2 shows the nomenclature scheme used in this paper. Branches are referred to as branch b_i, groups as $g_{i,j}$ and parameters as $t_{i,j}$.

3 The Basic Algorithm

The particular algorithm we have chosen is the one presented by Lozano-Perez in [12] due to its simplicity and intuitiveness. It first constructs the C space for the articulated figure. For the sake of completeness, the process is outlined below using our terminology.

If the manipulator has n links, its configuration space can be constructed as follows:

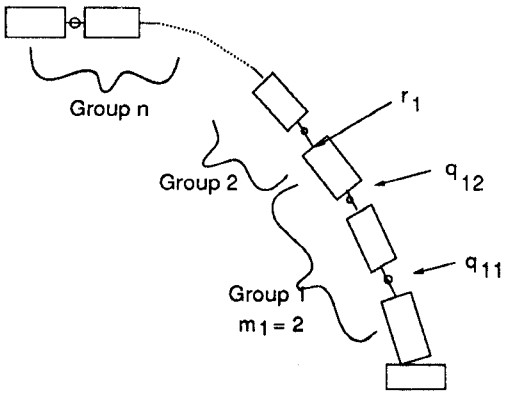

Figure 3: A Sequential linkage

1. $i = 1$.
2. While $(i < n)$ do
   (a) Ignore links beyond link i, and find the ranges of legal values of q_i by rotating link i around the position of joint i determined by the current value ranges of $q_1, ..., q_{i-1}$ and check for collision with the surrounding obstacles. If the link segment keep a list of segments of the same figure to avoid collision with, check the link segment with the segments in the list as well. This is for handling self collision in the figure. Mark as forbidden those joint values at which link i will have a collision.
   (b) Sample the legal range of q_i at the specified resolution.
   (c) Increment i and repeat step (a) for each of these valid range values for the next linkage.

The free space is then represented by the data structure called *regions* to explore the connectivity between the cells. A graph is then built on these region nodes and an A* search is conducted to search for a free path from the start node to the goal node.

4 The Sequential Algorithm

4.1 Overview

The *Sequential Algorithm* handles the motion planning problem for the Cgroups along a sequential branch. This algorithm is based on Gupta's work. We will discuss some of the differences in our work after presenting the algorithm.

4.2 The Algorithm

Referring to Fig. 3, let n be the total number of *Cspace groups* on this branch. Let the joint DOFs associated with the groups be represented by q_{ij} where i is the group number and j is from 1 to m_i where m_i is the maximum number of DOFs group i has. Let r_i be the reference vertex for group i. It is basically the distal vertex of the link associated with the DOFs in the group i. Let $r_i(t)$ denote the trajectory of the reference vertex r_i. The initial and goal configurations of the arm are given as q^s_{ij} and q^g_{ij}, i=1..n; $j = 1..m_i$.

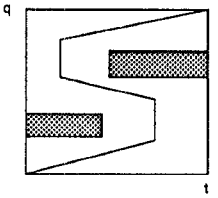

Figure 4: An example showing the case that a path can only be found with backtracking.

The algorithm is as follows:

1. Compute a collision free trajectory for the links associated with group 1. The trajectory of the reference vertex on its link will be $r_1(t)$.
2. $i = 2$.
3. While ($i < n$)
   (a) along $r_{i-1}(t)$, discretize the path according to a pre-specified resolution. Compute a collision-free trajectory for the DOFs in the ith group from q^s_{ij} to q^g_{ij} for $j = 1..m_i$ using the *basic* algorithm described in the last section.
   (b) given $q_{1j}(t), q_{2j}(t), ..., q_{ij}(t)$, compute $r_i(t)$ using forward kinematics.
   (c) Increment i.

4.3 Interpretation of the Parameter

The parameter used in parameterizing the path already computed can be either interpreted as temporal or non-temporal. For a temporal interpretation of the parameter, the path computed has to be monotonic with respect to the parameter t as we cannot travel backward in time. Hence backtracking is not allowed and the chance of finding a path is greatly restricted (Fig. 4).

In our system, we have adopted a non-temporal interpretation of the parameter in most cases as this will increase the chance of finding a path.

4.4 Discussion

Our implementation of the *Sequential* Algorithm differs from Gupta's in the following ways:

Gupta considered one DOF at a time in his work. We extend this idea further and introduce the notion of *Cspace groups*. Each Cspace group involves one parameter and a variable number of DOFs. Hence we are working on a higher dimensional C space.

The introduction of the Cspace groups has more far reaching effects than it appears at first sight:

- Theoretically, the *optimal* path (with respect to certain criteria, for example, shortest path or minimum energy) for an articulated figure consisting of n DOFs can only be obtained by searching through a n-dimensional C space built from

considering all n DOFs at one time. However, the well known exponential worst case time bound makes such algorithms impractical [15]. By considering only a subset of the DOFs at a time, we overcome the complexity problem but also reduce the degree of optimality of the solution. Apparently there is a spectrum of choices inbetween indicating a trade off between speed and optimality. The actual number of DOFs one can comfortably consider within a C group is usually dictated by the hardware available. In our implementation, we typically consider 2 to 3 DOFs within a group.

- The number of DOFs associated with the Cgroups can vary from group to group to fit into the structure of the figure being handled. For example, many joints within the human figure have 3 DOFs like the shoulder, waist, wrist. Hence, a 4-dimensional Cspace group is a very natural choice for these joints. The elbow joint, for example, may favor a 1-DOF Cgroup.

- Dealing with one DOF at a time also greatly limits the chance of successfully finding a path. Gupta has devised some backup strategies to handle difficult cases. Unfortunately, his technique only works for the first two links and there is no easy way to extend it. Dealing with more DOFs at a time have the advantage of adding more flexibility to the system. Backtracking is easier and more room is available for exploration.

Gupta has chosen to use the *visibility graph* for representing the free space and searching for a solution. A path found this way has the undesirable feature that the link associated with that group will be very close to some obstacles at certain points on the path. This will leave very little room for the next link to maneuver at those points and hence greatly reduce the chance of finding a free path for the succeeding Cspace group. Hence, we have opted to use the *region graph* instead which allows us to set the path closest to the center of the free space, thus allowing more room for the next link to maneuver (Fig. 5).

5 The Control Algorithm

5.1 Overview

The basic idea of our algorithm is to compute the motion for the linkage in each group in a particular traversal order. After computing the motion for each group, we parameterize the resulting path and pass on the parameter to be used in computing the path for the next group.

Since the articulated figure we are considering is essentially a tree structure, we need to adopt an order of traversal. We have chosen to adopt a depth first traversal, i.e. compute the motion for the branches one after another.

Finally, a special playback routine is needed to traverse the tree one more time and playback the final coordinated collision-free motion.

After we have found the free path for one branch, we need to record all the joint angles along this path in an array. The information is important in the final stage of playing back the free path as we will see in a later section.

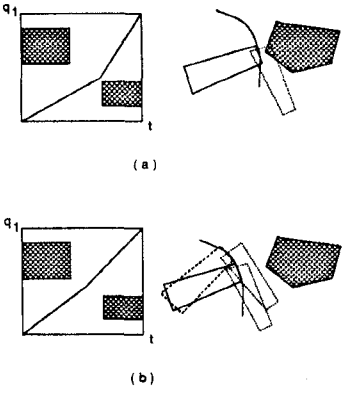

Figure 5: (a) A path found using visibility graph which is too close to an obstacle. This leave little room for the next linkage to maneuver. (b) A better path that is farther away from the obstacle found using region graphs.

There are some specific issues needed to be addressed in each of these stages. These are explained in details in the following sections.

Now let us look at the *control algorithm* itself before we look at the different issues involved.

5.2 The Control Algorithm (A1)

1. Apply Algorithm $ASUB1$ (the Planar Algorithm, see next section) to the whole figure to obtain the planar collision free translational movement of the figure taken as a whole.

2. Parameterize the resulting motion. The normalized constant for this motion is called $s_{initial}$.

3. Apply Algorithm $A2$ (the *Sequential* algorithm) to branch b_0 with $s_{initial}$ as the first parameter for the first group of the branch, i.e. group $g_{0,0}$.

4. Parameterize the resulting path computed for branch b_0 according to some prespecified resolution. The normalized parameter is named s_0. The trajectory of the reference point on branch b_0 is referred to as $r_0(t_1)$.

5. $i = 1$

6. While $(i < n)$ do

   (a) Apply Algorithm $A2$ to branch b_i. Use s_{i-1} as the parameter for the first group of this branch, i.e. group $g_{i,0}$.

   (b) Parameterize the resulting path with some prespecified resolution.

   (c) Invoke the Algorithm $A4$ (the Playback Algorithm) to branch b_i to obtain the sequence of joint angle values of branch b_i when moving along the computed path.

(d) Record this sequence of joint angle values in the array $FREEANGLES_i[1..N_{free}]$ where N_{free} is the total number of joint angle values recorded along this path.

(e) A normalized parameter, s_i, is defined to index into the array $FREEANGLES_i$ through the linear mapping:

$$Map(s_i) : \{0..1\} \rightarrow \{1..N_{free}\}$$

(f) Increment i

7. (Now $i = n$, the last branch) Apply Algorithm $A2$ (the *Sequential* algorithm) to branch b_n. Discretize the resulting path according to some resolution.

8. Apply Algorithm $A4$ (the playback algorithm) to the whole figure, starting from this very last group of the very last branch.

9. The angle values obtained can then stored into frames for continuous playback.

6 The Planar Algorithm (ASUB1)

The articulated figure can translate and rotate on a plane, navigating around obstacles. The whole figure behaves just like a mobile robot. We can handle this case simply with our *basic* algorithm (A3) or a separate module which will take advantage of customized mobile robot path planning techniques.

7 Resolving Conflicts between Different Branches

Although branches b_1 to b_n are attached to the same rear link of branch b_0, we do not use the same parameter t_{0,p_0} that parameterizes the motion of branch b_0 in all these branches. This is due to the fact that we have allowed the parameters t_{ij} to be interpreted as a nontemporal parameter (refer to last section). Hence backtracking is allowed and the values of t_{ij} along the computed path can be nonmonotonic. Therefore, some of the joint angle values cannot be obtained uniquely during the final playback phase.

Our solution to this problem is to further *parameterize* the already *parameterized path* of the previous branch. What this means is explained below.

Let us look at Fig. 6(a). The curved path shown in the top diagram represents the computed motion for branch b_0. The bottom diagram shows the resulting C space for the *last* group of branch b_1 (2-D for illustration). As we follow the computed path, it can be seen that the values taken by the parameter t_{1,p_1} along the path are not necessarily monotonic. We now choose to parameterize this path again before going on to compute the path for the next branch. It is this new parameter that we are using when dealing with

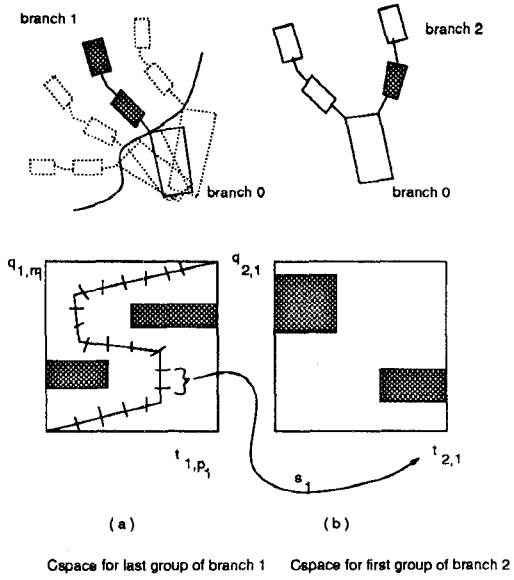

Figure 6: (a) Diagram above shows the parameterized path of the torso and the computed path for branch b_1. Diagram below shows the corresponding Cspace for the last group of branch b_1 (b) The Cspace and computed path for the first group of the branch b_1.

the first Cspace group of the next branch as shown in Fig. 6 (b).

The reason for this extra step is to generate a unique index into the computed joint angles. Since the playback algorithm traverses the branch in a backward manner, it needs an index that will give back an unique value for the computed joint angles. Without this reparameterization procedure, the index will be multivalued and the actual computed path cannot be determined during playback.

8 Playing back the free path

Now that the path for all the branches has been computed, the playback routine is called upon to playback the final path starting from the last group of the last branch.

For example, let Fig. 7 (a) represent the configuration space for the *last group* of the *last* branch, i.e. group g_{n,p_n} of branch b_n. We then discretize the computed free path according to a pre-specified playback resolution. Note that the number of discretization intervals for this last group will determine the total number of *time frames* for the final simulation.

At every discretized point along this path, say A, there is a corresponding (q,t) pair: the q value is what we should set the joint DOF to, and the *parameter* t is used to deduce the motion of the preceding (proximal) group. Note that within this preceding group, the parameter t is *monotonic* according to the way it is defined. Hence we can uniquely determine the corresponding (q,t) pair within this preceding group. With the same token, we can continue tracing back to the group further preceding this one (Fig. 7 (a)). We carry on in this fashion recursively until we come to the first

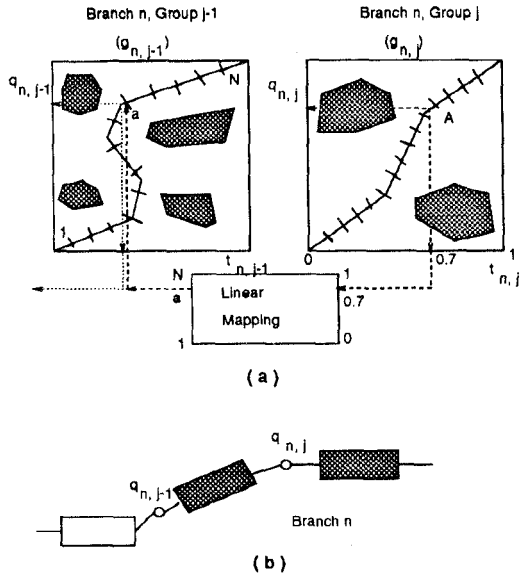

Figure 7: A figure showing how the final joint angle values of the whole figure are read off from the Cspace associated with the Cspace groups

group within this branch.

The sequence of joint values for moving all the other branches along their computed path should have been recorded in the array $FREEANGLES_i$ during the computation phase. The parameter value left unused is the first parameter of the first group of the last branch, i.e. $t_{n,1}$ which is actually the same as s_{n-1}. As discussed earlier, it is monotonic in values by the way it is constructed. Hence, we can use this as an index into the recorded joint angle array and uniquely determine the set of angles corresponding to the movement of the preceding branch. We then set those joint angles to the desired values in a somewhat *forward* manner.

After setting the angles in the preceding branch, all the other branches are treated in a similar manner.

We explain the other details of the algorithms by listing the complete playback algorithm in the following:

8.1 The PlayBack Algorithm (A4)

For easy understanding, we will explain the algorithm by looking into its two components separately: the Final Playback Algorithm (A4a) and the Single Branch Single Frame Playback Algorithm (A4b).

8.1.1 The Final Playback Algorithm (A4a)

This algorithm contains the control loop and will generate all the time frames for the simulation. The loop variable k can be thought of as the simulation frame index.

- Discretize the path computed for the last group in the last branch into N_{final} discrete points according to some pre-specified resolution.
- Let $k = 1$.
- While ($k < N_{final}$) do
  1. Apply Algorithm *A4b* (the Single Branch Single Frame Playback Algorithm) to branch b_n, the last branch in the figure to obtain the collision-free motion.
  2. The parameter value, $t_{n,1}$, will be obtained at the termination of the Algorithm *A4b*. We then use this parameter as an index into the array $FREEANGLES_{n-1}$. The joint angles recorded for branch b_{n-1} will be read off from the array element pointed to by this parameter value.
  3. Set the joint angles in branch b_{n-1} to the values read off from the array.
  4. Let $z = t_{n-1,1}$, the first parameter value of branch b_{n-1}, which is also read off from the array $FREEANGLES_{n-1}$.
  5. Let $i = n - 2$, the third last branch in the articulated figure.
  6. While ($i > 0$) do
     (a) Use the value of z as an index into the array $FREEANGLES_i$. Read off the joint angle values stored in the array elements.
     (b) Set the joint angles in this branch to the values obtained from the previous step.
     (c) Set $z = t_{i,1}$, the first parameter which is also read off from the array $FREEANGLES_i$.
     (d) Decrement i.
  7. Apply Algorithm *A4b* to the remaining branch, b_0, to get back its computed joint angles values and set the joints accordingly.
  8. The last parameter value obtained from the previous step is used to index into the path computed from the Planar Algorithm. The *position* of the whole figure will be set to that indexed position.
  9. Advance the simulation time step by incrementing k and repeat the whole playback process for the next time frame.

8.1.2 The Single Branch Single Frame Playback Algorithm (A4b)

This playback algorithm handles only the motion playback of one branch for only one time frame. Let the branch index we are considering be i. Here branch i has a total of p_i groups.

1. This algorithm only deals with one discretized point, and hence only one time frame. Let this discretized point be the kth point on the path.
2. Let $j = p_i$. j here is the group number index. We start from the last group in the branch and goes *down* the branch by decrementing j.
3. Let $r = k$. r is used as a loop variable to pass down the time frame index down the branch.

4. While ($j > 0$) do

   (a) From the rth discrete point on the computed path, read off the values of the $q_{i,j}$s from the q axis of the Cspace.

   (b) Set the corresponding joints in the articulated chain to the q values we have just found.

   (c) Then read off the normalized parameter value $t_{i,j}$ from the t axis.

   (d) Through a simple linear mapping, we can obtain the corresponding discretized point on the path computed for the previous group $g_{i,j-1}$ from this parameter value.

   (e) Set $r = index$

   (f) Decrement j

9 Recursive Application to the Whole Tree

The complete tree structure of the articulated figure can now be handled by the *control* algorithm described above. We need only to modify it to call itself recursively when it comes across another subtree during the tree traversal. The parameter it is currently worked with will be passed along with the recursive call. The playback routine will similarly call itself again recursively. We need to modify it to record the whole sequence of joint angles in a subtree after it has dealt with it. Then those angles can be recalled in the next level just like another branch as described before.

10 Results

Fig. 8 shows a human figure reaching through two holes with both arms. The path computed is collision free and involves more than 20 DOFs. Fig. 9 shows a more difficult example in which a bomb-discharge type of robot navigate a cluttered environment. This demonstrates the translational movement of the figure in addition to its revolute joints movement of the upper body. The examples are computed on a Silicon Graphics Personal Iris workstation under the *Jack* environment - a graphics environment developed for human figure modelling at PENN [14]. The first example finish in 9 minutes wall clock time, the second one in 17 minutes.

11 Discussions

The main advantages of our algorithm described here are that it can deal with articulated figures with branches and many degrees of freedom. The algorithm is an approximate algorithm, not a complete one, in the sense that it may not succeed to find a path even though there exists one. However, we can add a capability to backtrack among groups and branches. In this case, the algorithm will be a complete algorithm. Alternatively, we can try regrouping the DOFs under a different scheme and finding the solution again.

The *basic* algorithm within a Cspace group is $O(r^{k-1}(mn)^2)$ where k is the number of DOFs, r is the discretization intervals, m is the number of faces and edges for the robot and n for the environment as shown in [12]. Since the number of DOFs in a Cspace group is bounded, the run time for the *basic* algorithm can be treated as a constant. Consequently the whole algorithm runs in $O(p)$ time where p is the total number of groups in the tree structure without Cgroup backtracking. With Cgroup backtracking, the worst case time bound is exponential as with other complete algorithms.

We believe that the average run time of the algorithm is fast enough for practical use and that it will contribute to applications in robotics task level planning and human figure motion simulations.

References

[1] J. Barraquand, B. Langlois, and J. Latombe. *Numerical Potential Field Techniques for Robot Path Planning.* Technical Report STAN-CS-89-1285, CS Dept, Stanford University, 1989.

[2] J. Barraquand, B. Langlois, and J. Latombe. Robot Motion Planning with Many Degrees of Freedom and Dynamic Constraints. In *Fifth Intl. Sym. on Robotics Research (ISRR), Tokyo,* pages 1–10, 1989.

[3] J. Barraquand and J. Latombe. *Robot Motion Planning: A Distributed Representation Approach.* Technical Report STAN-CS-89-1257, CS Dept, Stanford University, May 1989.

[4] Rodney A. Brooks. Planning Collision-Free Motions for Pick-and-Place Operations. *Int. Journal of Robotics Research,* 2(4):19–44, Winter 1983.

[5] Rodney A. Brooks. Solving the Find-Path Problem by Good Representation of Free Space. *IEEE Transactions on Systems, Man and Cybernetics,* SMC-13(3):190–197, Mar 1983.

[6] Bruce Donald. *Motion Planning with six degrees of freedom.* Technical Report 791, MIT AI Lab, 1984. pp.504-512.

[7] Bruce Donald and Patrick Xavier. A Provably Good Approximation Algorithm for Optimal-Time Trajectory Planning. In *IEEE Intl. Conf. on Robotics and Automation,* pages 958–963, 1989.

[8] Bernard Faverjon. Obstacle Avoidance using an Octree in the Configuration Space of a Manipulator. In *IEEE Intl. Conf. on Robotics and Automation,* pages 504–512, 1984.

[9] Laurent Gouzenes. Strategies for Solving Collision-free Trajectories Problems for Mobile and Manipulator Robots. *Int. Journal of Robotics Research,* 3(4):51–65, Winter 1984.

[10] Kamal Kant Gupta. Fast Collision Avoidance for Manipulator Arms: A Sequential Search Strategy. *IEEE Transactions of Robotics and Automation,* 6(5):522–532, Oct 1990.

[11] Kamal Kant and Steven W. Zucker. Toward Efficient Trajectory Planning: The Path-Velocity Decomposition. *Int. Journal of Robotics Research,* 5(3):72–89, Fall 1986.

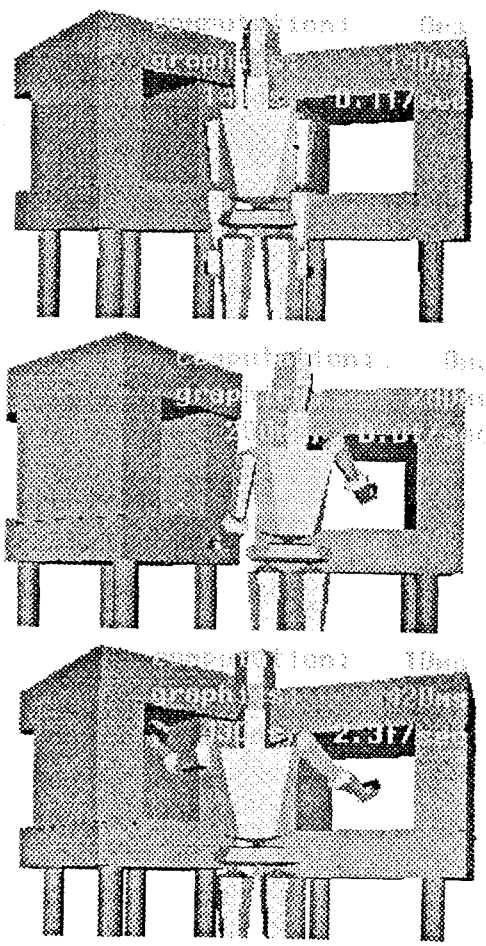

Figure 8: A human figure reaching through two holes and avoiding obstacles

[12] Tomas Lozano-Perez. A Simple Motion Planning Algorithm for General Robot Manipulators. *IEEE Journal of Robotics and Automation*, RA-3(3):224–238, June 1987.

[13] Tomas Lozano-Perez. Spatial Planning: A Configuration Space Approach. *Transactions on Computers*, c-32(2):26–37, Feb 1983.

[14] Cary B. Phillips and Norman I. Badler. JACK: A Toolkit for Manipulating Articulated Figures. In *Proceedings of ACM SIGGRAPH Symposium on User Interface Software*, Banff, Alberta, Canada, 1988.

[15] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: I. The case of a Two Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[16] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Advances in Applied Mathematics*, 4:298–351, 1983.

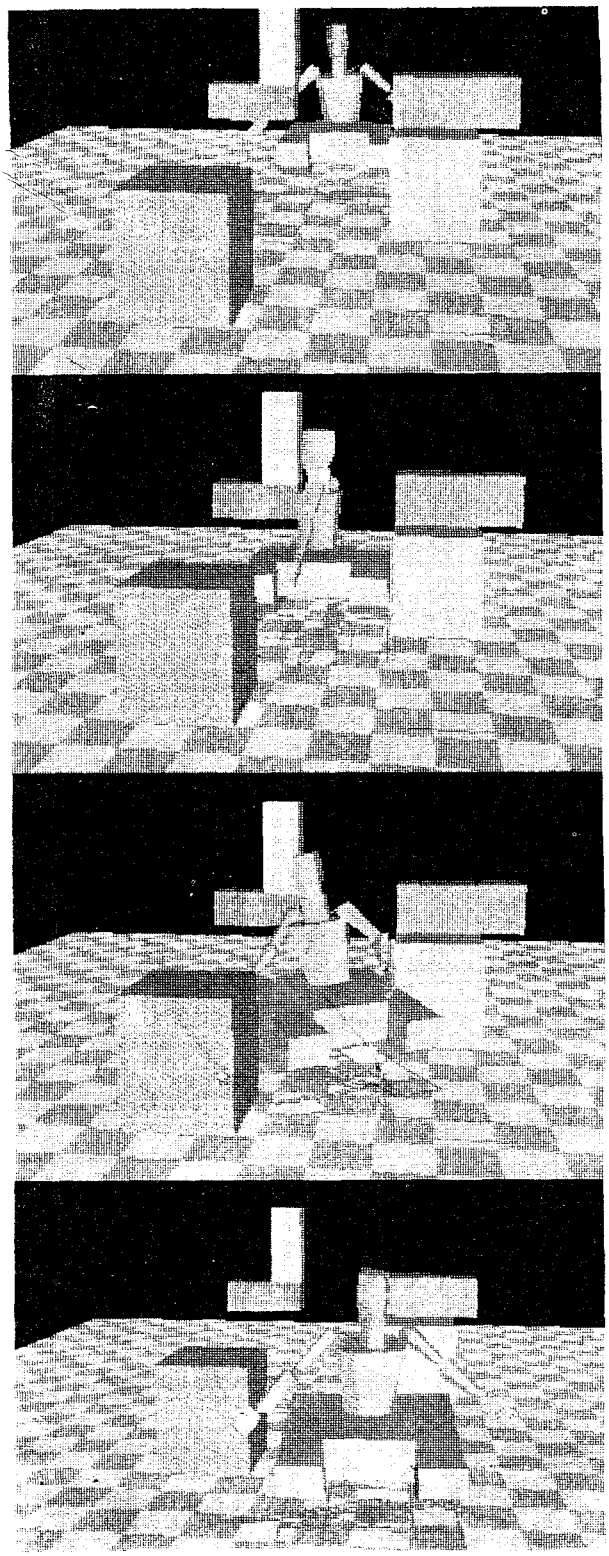

Figure 9: A bomb discharge robot executing collision free motion among a cluttered environment