*Appear in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2003*

# Motion Planning for a Crowd of Robots

Tsai-Yen Li
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.

Hsu-Chi Chou
Computer Science Department
National Chengchi University,
Taipei, Taiwan, R.O.C.

***Abstract* - Moving a crowd of robots or avatars from their current configurations to some destination area without causing collisions is a challenging motion-planning problem because the high degrees of freedom involved. Two approaches are often used for this type of problems: *decoupled* and *centralized*. The tradeoff of these two approaches is that the decoupled approach is considered faster while the centralized approach has the advantage of being complete. In this paper, we propose an efficient centralized planner that is much faster than the traditional randomized planning approaches. This planner uses a hierarchical sphere tree structure to group robots dynamically. By taking advantage of the problem characteristics on independently moving robots, we are able to design a practical planner with the centralized approach when the number of robots is rather large. We use several simulation examples to demonstrate the efficiency and effectiveness of the planner.**

## I. INTRODUCTION

The problem of directing a fleet of robots or moving a crowd of avatars are often raised in the context of robot contest, computer animation, and simulation for urban planning. The problem is challenging because the high degrees of freedom involved when the number of robots becomes large. The curse of dimensionality makes the problem difficult to solve [16]. Generally speaking, there are two main approaches to the planning problem for multiple robots: *decoupled approach* and *centralized approach*. The tradeoffs between these two approaches lie on efficiency and completeness. The decoupled approach is typically faster but lacks completeness while the centralized approach can be made complete but might need a large amount of planning time and storage space.

When the degrees of freedom in a system are rather independent in nature, the decoupled approach might be a good solution since the planning time for each decomposed problem could be rather short. The algorithm developed for solving a simple subproblem can also be complete. However, when the planner for the decomposed subproblem fails, there are usually no good algorithms that can backtrack and systematically try alternative decomposition. If we choose to use a centralized approach to solve the problem, a complete method can be developed to search the composite configuration space systematically. However, since the size of the composite configuration space is overwhelming, a systematic search dooms to be impractical. Therefore, most planners with the centralized approach use a randomized algorithm to achieve probabilistic completeness.

Although randomized algorithms have been shown to be a practical approach to solve motion-planning problems in high dimensional configuration space, we found that they may fail to find a feasible path when the decoupled degrees of freedom are actually interfering with each other such as in the case of robot crowds. In this paper, we propose a novel centralized planning approach that moves the robots in groups formed dynamically with a sphere-tree structure. We use several examples to demonstrate that the traditional randomized path planners fall short when the number of robots becomes large. With the new approach, on the other hand, we can plan for a larger number of robots in a more efficient way. We have also implemented a decoupled planner to demonstrate that the centralized planner could be a better choice in terms of completeness and efficiency.

The rest of the paper will be organized as follows. We will review the related work on the planning problem for multiple robots in the next section. In the third section, we will describe the basic problem and present an implemented planner with a decoupled approach. Then, we will propose our new centralized approach in Section IV. In Section V, we will use some experimental data to demonstrate the effectiveness of our approach. Finally, we will conclude the paper in the last section.

## II. RELATED WORK

Surveys of motion planning algorithms can be found in [11] and [7]. According to [7], path planning can be viewed as either *centralized* or *distributed*. The centralized planning typically considers all robots and their degrees of freedom altogether and therefore usually entails a high dimensional composite search space. In a distributed approach, each individual robot plans and adjusts its paths in parallel with other robots until feasible paths for all robots are found.

In [11], the taxonomy about planning for multiple robots is somewhat different. The approaches are classified into two categories: *centralized* and *decoupled*. The decoupled planning is different from the distributed planning on that the robots are planned sequentially in the decoupled approach. Two variations exist in decoupled planning: (1) *prioritized planning* that considers one robot at a time under the constraint of previously planned paths for other robots, (2) the *path coordination method* that schedules the execution of individually planned paths to avoid interference. The work of [6] is an example of decoupled approach where all robots are prioritized and planned with respect to only higher-priority robots. A similar approach has also been proposed in [14] to generate the motions of two manipulator robots in an on-line manner. In [13], a decoupled approach has been used to generate motions for avatars in a virtual environment.

Most methods originally developed for single-robot systems can be applied in centralized planning. However, due to the high dimensionality of such a system, a complete planner is usually intractable. In the past decade, the randomized planning approach [1] has attracted much attention and been successfully demonstrated in many applications with difficult problems [5][10]. Early randomized planners use artificial potential fields built in the workspace to guide the search in C-space and use random walks to escape local minima. A typical planner with this approach is the RPP (Randomized Path Planner) [2]. Most of the randomized planners developed in the last few years use the PRM (Probabilistic Roadmap Method) approach [9]. In such an approach, we build a random roadmap in the C-space in a preprocessing step and try to answer planning queries at a later time as quickly as possible. A common feature of the randomized planners is that they are probabilistic complete, which means that if there exists a feasible path and there is no time limits, then the planner will be able to find it eventually.

The research on generating motions for crowds of agents can be found in the literature of Robotics, Artificial Life, and Computer Animation. A good survey of cooperative robotics can be found in [3]. In [8], a flocking model was used for a crowd of robots to follow a leader robot. A similar approach has been adopted in [12] to simulate a crowd of avatars led by a leader capable of generating collision-free motions. Realistic flocking behaviors for virtual creatures such as birds or fishes have been successfully simulated with artificial forces [17]. In [3], roadmap consisting of medial axes is used to guide the simulation for a flock of avatars. However, a common weakness of these approaches is that they cannot guarantee that a feasible motion plan can be generated for the whole system even if such a plan exists.

## III. THE DECOUPILED PLANNING APPROACH

In this section, we will first give a general description of the path-planning problem for multiple robots. The problem definition, in fact, might be different for different applications at various situations. However, we will focus on the problem suitable for the decoupled approach in this section and briefly describe a planner implemented with this approach. Examples generated with this approach will be given at the end.

### *A. Considerations for Different Applications*

Depending on the applications, one can define the planning problem for multiple robots slightly differently. For example, depending on the time when the problem is raised, a planner may need to plan for all robots at a time or it may be called sequentially for each robot when their paths are needed. For the first case, the problem can be solved with either a decoupled or a centralized approach. However, for the second case, a decoupled planning is more appropriate since the path for each robot is generated at different time.

Another application attribute that might affect the choice of the planning approach is the specification of the goal configuration. If each robot can be given a definite goal configuration at run time, we can use either approach to solve the problem. However, specifying the goal configuration for a large number of robots is a tedious task. If we must generate the motions for all robots at a time, we are more likely to specify a rough destination region for the robots instead of individual goals. If the destination region is not very large, the decoupled approach may not be a good choice because the robots that reach the region earlier are likely to block the entrance and prevent later robots to reach the region. In the next subsections, we will assume that the planning requests are issued at run time while the motions of other robots are being executed, and each robot will be given a specific goal configuration.

### *B. Motion-Planning Problem for Multiple Robots*

Assume that we are given a geometric description of the robot and polygonal obstacles in a 2D workspace. We assume that each robot can be represented by an enclosing circle of radius $r$. Due to geometric symmetry, we can use only two parameters ($x$, $y$) to describe the configuration $q^i$ for a robot $i$. Suppose that there exist $n$ robots ($n$>1) in the workspace. We denote the individual configuration space (C-space) for robot $i$ by $C^i$. Then the composite configuration space for the multi-robot system is defined as $C= C^1$ x $C^2$ x…x $C^n$, where a configuration in $C$ is denoted by $q$. Each robot has to satisfy the geometric constraint that they cannot collide with each other or with obstacles. In addition, each robot must move under a velocity limit constraint.

In this section, we will assume that the planning requests are issued at different times for different robots while the motions of other robots are being executed. Each request defines a planning problem for a robot from its current configuration to a specified goal configuration. Although not necessary, it is usually desirable not to disturb the current motion plans of other robots when we try to find a feasible motion for a robot. Therefore, a decoupled approach is more appropriate for this case.

### *C. Decoupled Planning Approach*

Assume that we are given a motion-planning problem for multiple robots as described in the previous subsection. The path of the $i$th robot (denoted by $\tau^i$, $i$ = 1 to $n$) is known as a function of time $t$, including when it is static. In our decoupled approach, for the $k$th robot under consideration we augment its C-space by the time dimension to form the so-called *Configuration-Time Space* (*CT-space*). A conceptual example is depicted in Fig. 1. There are two types of

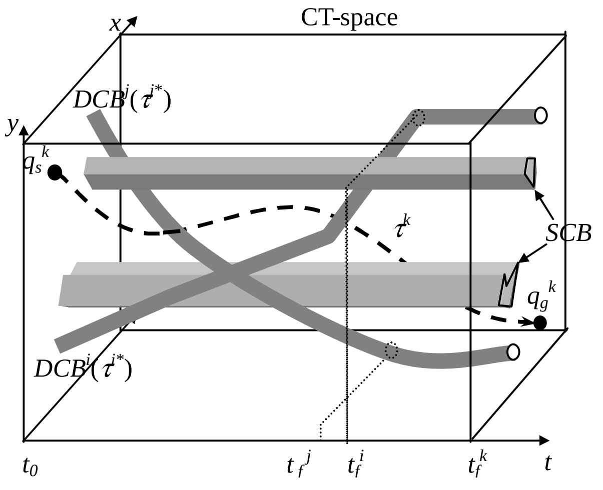


Fig. 1. Searching for a feasible path amongst obstacle regions in the CT-space.

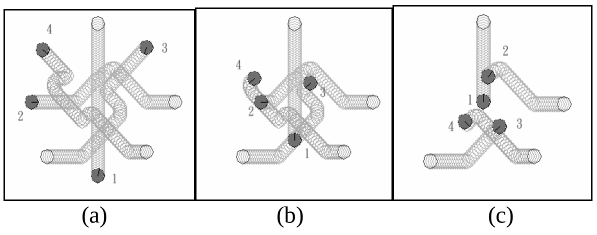


Fig. 2. An example of decoupled planning with coordinated crossing motions for four robots

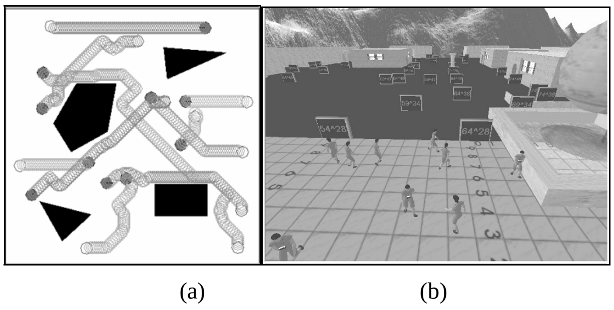


Fig. 3: An example of decoupled planning for a crowd of avatars moving independently in a virtual world

forbidden regions in the CT-space representing obstacle regions that the robot should avoid entering. One (denoted by *SCB*) is due to the static obstacles while the other (denoted by *DCB*) is due to other moving robots. Note that *SCB* is axis-parallel extrusion of 2D obstacles in time while *DCB*'s are curve extrusions of the obstacle regions imposed by other moving robots. When the *i*th robot finishes its motion at time $t_f^i$ , we assume that it will stay there unless otherwise instructed. Equivalently, we are extending the path for the *i*th robot to infinity and this extended path is denoted by $\tau^{i*}$. The objective of the path planner is to find a collision-free path $\tau^k$ for the *k*th leader in the CT-space that can connect the current ($q_s^k$) configuration at the current time ($t_0$) to the specified goal configuration ($q_g^k$) at some time ($t_f$) in the future. Because of the velocity constraint the slope at any point along a legal path in this CT-space must be positive (because time is not reversible) and less than some user-specified value (maximal velocity). A Best-First planning algorithm can be adopted to search for a feasible path in such a CT-space.

### D. Planning Examples

Fig. 2 shows an example of decoupled planning for four robots moving across each other. The trace of their paths is shown, and the planning order (priority) is depicted beside the robots. Note that the first robot chooses a straight-line path since it has the highest priority. The later a robot is planned, the more detoured its path usually will be. Another example of multiple robots moving independently without colliding with each other is shown in Fig. 3(a). In Fig. 3(b), we show a snapshot of how the planner has been used to simulate a human crowd in a virtual environment.

In the example of Fig. 2, the average planning time for each robot is about 105ms on a regular PC with 650MHz CPU. When the number of robots increases, one can expect that the planning time will have a quadratic growth because a robot has to check collisions with other $n$-1 robots. Although the growth is not as fast as the exponential growth in the centralized approach, the advantage that the decoupled approach can be used in an on-line manner for interactive applications may not be valid when $n$ increases to some large value.

## IV. THE CENTRALIZED PLANNING APPROACH

### A. Revising Problem Definition

When the number of robots increases to some large value, say 100, it becomes impractical to specify the goal configuration for each individual robot interactively. In this case, it is more desirable to specify a rough destination region for the crowd of robots to move to. We assume that the region is a circle of radius $R$ centered at ($x_g$, $y_g$), specified by the user. The goal is reached if all robots can enter the region enclosed by the circle. We assume that the order of entering and their relative positions are not important. Although a decoupled approach can be used to solve the problem but such an approach often fails to find a path simply because the robots that arrive early may prevent later robots to enter the region. Therefore, a centralized approach is preferred. However, one has to face the curse of dimensionality as the number of robots increases.

Among the randomized path planning algorithms proposed in the literature, the RPP (Randomized Path Planner) and PRM (Probabilistic Roadmap Method) are the two mainstream methods. However, both methods share some

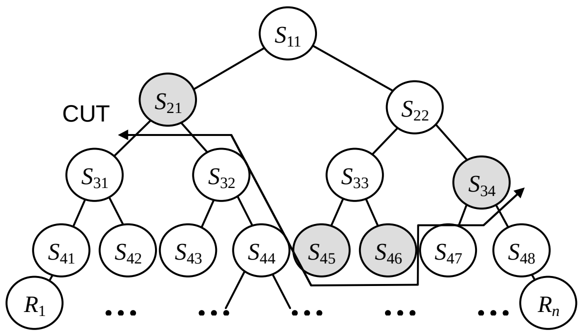


Fig. 4. A sphere tree and a profile cut separating spheres with and without collisions

common characteristics and have their own pathological cases [1]. For example, the PRM planners typically have difficulties in connecting two roadmap components through long narrow passages. The RPP planner could be better in solving difficult planning problems but it typically takes a longer time when the heuristic potential fields used in the planner are misleading.

Although the motion-planning problem for multiple robots can be solved with either centralized approach, we think pathological cases often occur in the traditional planners as the number of robots increases. When the number of robots is large, it is more likely that the robots are rather crowded at the initial and goal configurations. In this case, the probability of finding a legal neighboring configuration becomes rather low since the robots all move independently. As long as one robot is in collision, the overall system configuration becomes illegal. Therefore, the size of solution space is rather small compared to the whole problem space when we allow every robot to have its full degrees of freedom. The set of legal configurations would be limited to those that move the robots at the periphery outwards first. However, the probability of choosing such a configuration is rather low in a random process. Therefore, we need to improve the traditional planner by accounting for this problem characteristic.

### *B. Grouping Robots with a Hierarchical Sphere Tree*

As described in the previous subsection, when we plan for multiple independent robots, the pathological case happens because we are giving the robots too much freedom. Allowing only a few robots to move at a time may be a good idea but the planner may not be probabilistic complete any more. In addition, one still has to determine who to move first. Therefore, we propose to organize the crowd of robots into a hierarchical sphere tree structure and move the robots as a set of robot groups whenever possible. A sphere tree is a binary tree, whose leaf nodes represent geometric primitives, such as a circle or a sphere, composing the shape of a robot. Each internal node represents a sphere whose size is large enough to enclose its children spheres. This type of sphere tree structure is commonly used to reduce the number of calls to expensive collision detection routines [15]. As long as the bounding volume of a node at a higher level does not cause collisions, further examination below the node becomes unnecessary.

```
procedure Down_Motion_with_Grouping()
1. SUCCESS : = false
2. Append( qi, $\tau$ )   { $\tau$ is the path for down motion }
3. nStep : = 0   { nStep is number of legal moves }
4. CUT : = root   { a profile cut list of the sphere tree }
5. while ¬ SUCCESS
6.     nTrial : = 0   { nb of trials for lower legal neighbors }
7.     nTotalTrial : = 0   { nb of trials for local minima }
8.     while nTrial < nMaxTrial
9.         nTrial : = nTrial + 1
10.        nTotalTrial : = nTotalTrial + 1
11.        for all spheres o in CUT   { o is a sphere }
12.            o' : = SelectLegalNeighbor(o)   { random select }
13.            if o' = NULL then Split(o, CUT) { split o}
14.        $q_{new}$ : =  Conf(CUT)   { find corresponding conf }
15.        if Legal($q_{new}$) then   { collision-free or not }
16.            nTrial : = nTrial + 1
17.            if $U(q_{new}) < U_{min}$ then   { lower potential }
18.                Append($q_{new}$, $\tau$ )
19.                nStep : = nStep + 1
20.                break   { while }
21.        else if Crowded() then RebuildST()
22.    if nTrial >= nMaxTrial then SplitLargestSphere(CUT)
23.    if nStep mod nPeriod = 0 then RebuildST()
24.    if $U_{min}$ = 0 then SUCCESS : = true
25.    if nTotalTrial > nMaxTotalTrial then break {local min.}
```

Fig. 5: The Down_Motion_with_Grouping algorithm

We build a sphere tree for the robots at their initial configuration. The robots are organized in a hierarchical structure where each leaf node represents a robot, as shown in Fig. 4. Since each leaf node belongs to a list of ancestor sphere nodes of various sizes, robots can be grouped and moved with different levels of grouping. When we move an internal sphere node, all robots under the node also move for the same amount. When a sphere node moves, the ancestor spheres up to the root must update their radius accordingly in order to enclose their children nodes. This is somewhat different from the case of pure collision detection applications where the relative positions between spheres in a sphere tree do not change because most applications assume that the robot is a rigid body.

### *C. RPP with Hierarchical Grouping*

The low probability of moving to a legal neighbor makes the planning problem a pathological case for path planners, especially for PRM-based planner. Therefore, we have chosen to improve the potential field based planners (RPP) by incorporating a hierarchical grouping strategy to increase the chance of finding a legal neighbor. The RPP algorithm consists of alternative calls to the Down_Motion and Brownian_Motion procedures. The modifications that we

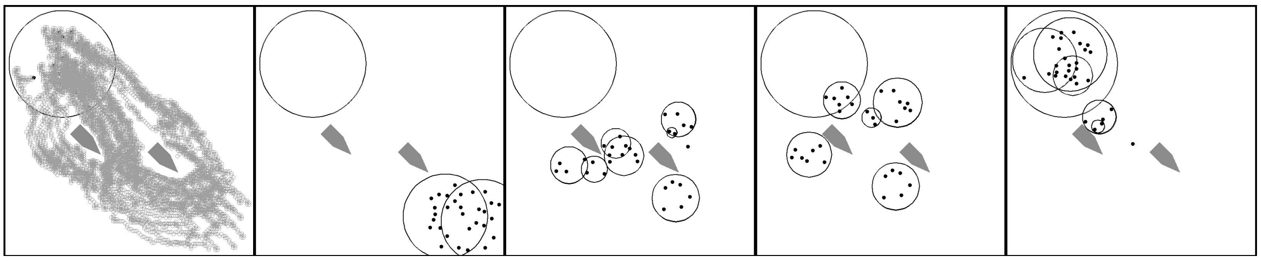


Fig. 6: An example showing how the planner dynamically changes robot grouping to reach the goal

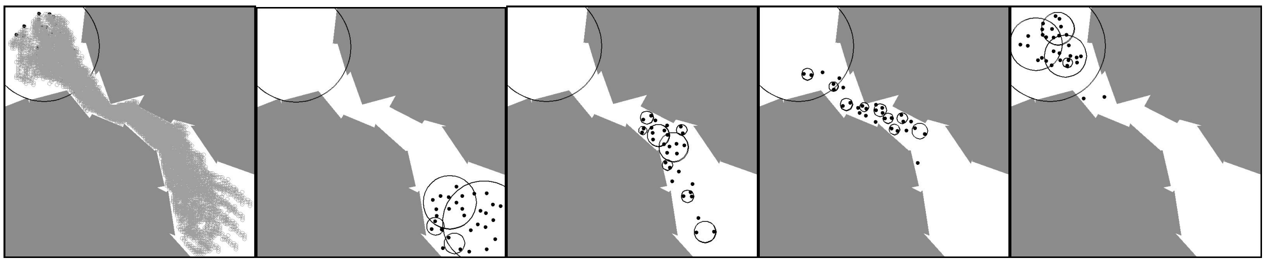


Fig. 7: An example showing a crowd of 30 robots passing a narrow passage

have made are mainly on using the grouping strategy to generate legal neighbors in the Down_Motion procedure.

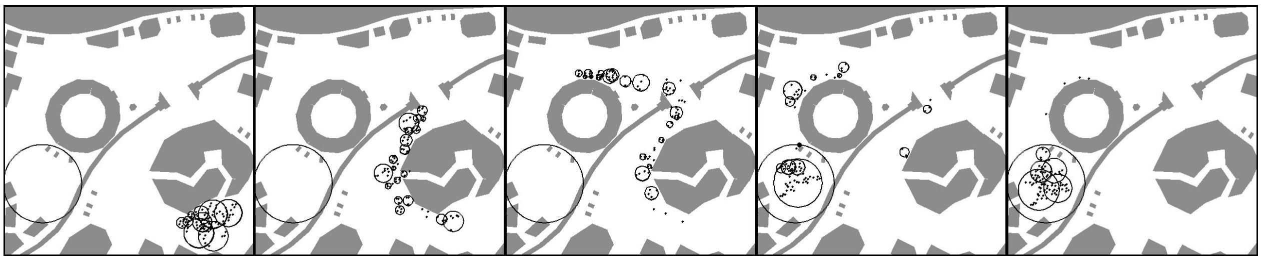
In Fig. 5, we show the modified procedure, called Down_Motion_with_Grouping. The idea is to freeze the relative positions between robots as much as possible by grouping them with hierarchical spheres as described in the previous subsection. The grouping attempts start from the root of the sphere tree and walk down toward the leaf nodes when the current grouping sphere collides with obstacles. When testing the possibility of grouping robots at an internal node, we randomly try a few neighboring configurations for the sphere (line 12) until a legal (collision-free with obstacles) configuration has been found or all trials fail. When the attempt fails, we will recursively walk down to the next level and attempt to move its two children spheres independently. When a legal configuration for the whole system has been found, we will have a list of grouping spheres (depicted in grey in Fig. 4) at various levels that forms a profile cut on the sphere tree. We will record this "CUT" location and start the next trial from it. In addition to considering the collisions with obstacles, we also have to check the inter-collision between robots (line 15). If the new system configuration is legal, then we check if the new configuration has a smaller potential value than the current one. If so, we will move the robots to the new configuration, and the search for the next legal configuration with a lower potential starts over again.

We set a limit on the number of trials for moving the robot system to a configuration with a lower potential value. If the number is reached, we further lower the CUT by splitting the largest sphere (line 22). This step enables the CUT to move to the lowest level (leaves) and restores the freedom of each robot. However, if we can only lower the CUT, the advantage of grouping the robots will disappear eventually when all robots retain their freedom. Therefore, we also have to consider moving the CUT upward as well. However, according to [12], attempting to merge nodes in every step from bottom up might not be more efficient than updating the list from the root node down after a few steps. In addition, when two sibling spheres move away from each other, the radii of their parent spheres may increase to a degree that rebuilding the sphere tree is desirable. Therefore, we periodically rebuild the sphere tree (line 23) and update the CUT from the top down to maintain a more representative sphere tree for future uses. In addition, when we detect that the robots are away from the obstacles but the inter-collision between robots are severe (tested in the Crowded function in line 21), we also choose to reorganize their relative positions by rebuilding the sphere tree.

We build a numerical potential field [2] in the C-space of a robot to guide the search. Since the goal is a destination region instead of a single configuration, we set the potential values of all configurations in the region to zero. The overall potential $U(q)$ for a system configuration $q$ is the sum of the potentials for each individual robot. When all robots enter the region, $U$ will become zero. If the system has made a given number (nMaxTotalTrial) of trials to move to a lower potential without success, we will assume that a local minimum has been reached and the procedure will return its current available path found so far. A random walk will be used to escape the local minimum as in the traditional RPP.

## V. EXPERIMENTS

We have implemented the RPP planner with dynamic hi-



(a) (b) (c) (d) (e)

Fig. 8: An example of centralized planning for a crowd of 80 robots moving to a destination region

Table 1. Comparisons of the planning times (in seconds) for moving different number of robots in three different planners

| N | Decoupled | Centralized RPP w/o Sphere Tree | Centralized RPP w/ Sphere Tree |
|---|---|---|---|
| 10 | 6.67 | 0.93 | 0.28 |
| 20 | 15.29 | 2.85 | 0.72 |
| 30 | 23.60 | 6.76 | 1.21 |
| 40 | 29.52 | 13.45 | 1.51 |
| 50 | 43.03 | 26.49 | 3.36 |
| 60 | 62.49 | 47.84 | 3.71 |
| 70 | 89.10 | 70.28 | 4.46 |
| 80 | 122.87 | 110.39 | 13.40 |
| 90 | 127.44 | 175.04 | 17.73 |
| 100 | 204.683 | 239.59 | 27.99 |
| 120 | 275.21 | 447.09 | 56.84 |
| 140 | 526.91 | 810.07 | 71.15 |
| 160 | 2224.05 | 1570.83 | 170.89 |
| 180 | --- | 2359.68 | 262.44 |
| 200 | --- | --- | 309.91 |
| 220 | --- | --- | 784.93 |
| 240 | --- | --- | 1594.19 |
| 260 | --- | --- | 1365.34 |
| 280 | --- | --- | 2502.34 |

erarchical grouping in the Java language. We have conducted extensive experiments to demonstrate the efficiency of the planner. All experimental data reported in this section were measured on a regular PC with 1.2GHz CPU.

### A. Planning Examples

In Fig. 6 and Fig. 7, we show the example paths generated by the planner for two different workspaces. In both examples, there exist thirty robots trying to move from their configurations to a destination region depicted in circle. The first subfigure(a) in both examples shows the traces of the paths executed by the robots. Note that in the example of Fig. 6, the robots move together as large groups until they encounter obstacles. In this case, the robots are organized into smaller groups and resume more degrees of freedom. The sphere tree may be rebuilt at run time so that we can see the robots are grouped differently in Fig. 6(d). As some robots reach the destination region, they still have some degrees of freedom to move inside the area so that they do not block the entrance where they entered. Fig. 7 shows an example in another workspace where there exists a narrow passage that forces the robots to move individually as they pass the passage. This is a pathological case for the planner since the advantage of moving robots in groups disappears and we still have to pay the cost of maintaining the sphere tree. However, since we rebuild the sphere tree periodically, the planner can resume moving in groups as soon as they pass the passage. We found that the planner with dynamic grouping is still more efficient than the traditional RPP in this example, which implies that the overhead of maintaining the sphere tree is rather low.

Table 2. The planning times (seconds) for using different methods to rebuild the sphere tree

| Methods | (A) no rebuild | (B) when crowded | (C) periodic | (D) periodic and crowded |
|---|---|---|---|---|
| Planning time (sec.) | 93.25 | 53.84 | 5.47 | 4.88 |

### B. Performance Comparisons

We have conducted extensive experiments to compare the performances of the decoupled planner, the traditional RPP planner and the new planner. The results are shown in Table 1. We run the three planners (decoupled, centralized with and without using sphere tree to group robots) for the workspace shown in Fig. 8. Snapshots along an example path generated for 80 robots are shown in Fig. 8. The number of robots ranges from 10 to 300 in the experiment. All planners are forced to terminate when the planning time exceeds one hour (marked with '---' in Table 1). Note that the new planner outperforms the decoupled and the traditional planners in all cases and the more the number of robots, the more improvement that we can observe. Although the traditional RPP planner is probabilistic complete, it is terminated after one hour of trial when the number of robots reaches 200. On the other hand, the new planner can still find a path when the number of robots reaches 280 (or 560 DOF).

We also have conducted experiments to study the effects of rebuilding the sphere tree that keeps its size small. In two ways, we rebuild the sphere tree. One is by detecting the situation that most collisions occur between robots instead of between robots and obstacles while the other is by periodic updates. The results are shown Table 2. The cases (B) and (C) correspond to the two methods above. Note that timely rebuild of the sphere tree can improve the overall performance and the effect of periodic updates seems to be more significant than the other case.

### *C. Discussions*

The improvement of the new planner over the traditional RPP planner is quite significant. Although we are not attempting to deal with the curse of dimensionality, we have significantly lowered the constant of the exponent to make the planner practical when the number of robots is large. The centralized planner outperforms the decoupled planner in most cases when the number of robots is not large. Besides, the centralized planner has the advantage of being probabilistic complete that the decoupled planner does not have. Detail data from our experiments show that the new planner with dynamic grouping is more efficient mainly because the number of inter-robot collisions has been significantly reduced. This observation reveals that the original idea of using a hierarchical sphere tree to group robots dynamically in order to reduce inter-robot collisions is quite effective.

## VI. CONCLUSIONS

The problem of path planning for multiple robots is getting more attentions in Robotics and Computer Animation. However, the traditional planners do not seem to be able to solve the problem as efficiently as in other cases. In this paper, we reviewed the different approaches proposed in the literature, and implemented the planners with these approaches for comparisons. We also have proposed a new planner based on the RPP planner to improve the planning performance for a large number of robots. Experiments show that this new method can significantly reduce inter-robot collisions and therefore is more effective for this type of multiple-robot planning problem.

## VII. ACKNOWLEDGMENTS

This work was partially supported by National Science Council under contract NSC 91-2213-E-004-005.

## VIII. REFERENCES

[1] J. Barraquand, L. Kavraki, J.C. Latombe, T.Y. Li, and P. Raghavan, “A Random Sampling Scheme for Path Planning,” *Intl. J. of Robotics Research*, 16(6), pp.759-774, Dec. 1997.

[2] J. Barraquand and J. Latombe, “Robot Motion Planning: A Distributed Representation Approach,” *Intl J. of Robotics Research*, 10:628-649, 1991.

[3] O. B. Bayazit, J.M. Lien, N. M. Amato, “Simulating Flocking Behaviors in Complex Environments,” *Proc. of the Pacific Conf. on Computer Graphics and Applications*, 2002.

[4] Y.U. Cao, A.S. Fukunaga, A.B. Kahng, and F. Meng, “Cooperative Mobile Robotics: Antecedents and Directions,” in *IEEE/TSJ Intl. Conf. on Intelligent Robots and Systems*, pp.226-234, 1995.

[5] H.S. Chang and T.Y. Li, "Assembly Maintainability Study with Motion Planning," *Proc. of 1995 IEEE Intl. Conf. on Robotics and Automation*, Nagoya, Japan, May 1995.

[6] M. Erdmann and T. Lozano-Perez, “*On Multiple Moving Objects*,” AI Memo No. 883, Artificial Intelligence Laboratory, MIT, 1986.

[7] K. Fujimura, *Motion Planning in Dynamic Environments*, Springer-Verlag, New York, 1991.

[8] V. Gervasi and G. Prencipe, “Flocking by a Set of Autonomous Robots,” Technical Report: TR-01-24, Department of Information, University of Di Pisa, Italy, 2001.

[9] L. Kavraki, P.Svestka, J. Latombe, and M. Overmars, “Probabilistic Roadmaps for Fast Path Planning in High-Dimensional Configuration Spaces,” *IEEE Trans. on Robotics and Automation*, 12:566-580, 1996.

[10] Y. Koga, K. Kondo, J. Kuffner, and J.C. Latombe, “Planning Motions with Intentions,” *Computer Graphics (SIGGRAPH’94)*, pp.395-408, 1994.

[11] J. Latombe, *Robot Motion Planning*, Kluwer, Boston, MA, 1991.

[12] T.Y. Li and J.S. Chen, “Incremental 3D Collision Detection with Hierarchical Data Structures,” in *Proc. of ACM Symp. on Virtual Reality Software and Technology, (VRST’98)*, pp.139-144, Taipei, Taiwan, 1998.

[13] T.Y. Li Y.J Jeng, and S.I Chang, “Simulating Virtual Human Crowds with a Leader-Follower Model,” *Proc. of the 2001 Computer Animation Conf.*, Korea, 2001.

[14] T.Y. Li and J.C. Latombe, “Online Manipulation Planning for Two Robot Arms in a Dynamic Environment,” *Intl. J. of Robotics Research*, 16(2):144-167, 1997.

[15] S. Quinlan, "Efficient Distance Computation between Non-Convex Objects," *Proc. of Intl. Conf. on Robotics and Automation*, pp.3324-3329, San Diego, CA, 1994.

[16] J.H. Reif, “Complexity of the Mover's Problem and Generalizations,” *Proc. of the 20th IEEE Symp. on Foundations of Computer Science*, pp. 421-427, 1979.

[17] C. Reynolds, “Steering Behaviors For Autonomous Characters,” *Proc. of Game Developers Conf.*, 1999.