On Multi-Arm Manipulation Planning

Yoshihito Koga and Jean-Claude Latombe
Robotics Laboratory, Department of Computer Science, Stanford University
Stanford, CA 94305, USA

Abstract: *This paper considers the automatic generation of motion paths for several cooperating robot arms to manipulate a movable object between two configurations among obstacles. To avoid collisions the robots may have to change their grasp of the object, for example, by passing it from one arm to another. The case where the movable object can only be moved by two arms acting simultaneously is also considered. An approach for solving this planning problem is described and illustrated with a robot system made of three arms moving in a 3D environment. Experiments with a planner implementing this approach show that it is not only fast, but also reliable in finding collision-free paths.*

Acknowledgments: This research was funded by ARPA grant N00014-92-J-1809. Y. Koga is supported in part by an NSERC fellowship. This research benefited from comments by Tsai-Yen Li and Gerardo Pardo-Castellote.

1 Introduction

Consider the task of programming a *single* robot arm to open a water faucet, and suppose two full revolutions of the handle are required. The robot must move and grasp the handle along a collision-free path. Then it must rotate the handle in the required direction. During this rotation, one joint of the arm will reach its limit, consequently requiring the robot to ungrasp the handle and regrasp it such that again it can be rotated. This process must be repeated until the two full revolutions are completed. Finally, the robot should ungrasp the handle and move back to some home position. Manipulation planning is the automatic generation of such a sequence of motion paths that delivers one or several movable objects (the faucet handle in this example) to a given goal configuration. This sequence is called a manipulation path. A crucial aspect of manipulation planning, relative to more classical path planning, is that it must account for the robot's ability to change its grasp of an object.

Manipulation planning becomes even more challenging when the robot system contains *multiple* arms. This brings the additional difficulty of deciding which arm(s), at any one time, must grab and move a movable object and how the various arms should cooperate along the manipulation path to ensure the delivery of the object to its goal configuration. Regrasping the object may now involve changing the arms grasping it. We call this the multi-arm manipulation planning problem.

There is much motivation for the study of automatic multi-arm manipulation planning. Multi-arm systems can be significantly more efficient than single-arm systems, e.g., by performing several motions simultaneously. They can also accomplish a greater variety of tasks. For example, in addition to the arms working independently of each other, they can cooperate to manipulate heavy and/or bulky objects by sharing the load for fast and responsive motions. One can also increase the workspace of the movable objects by having the robots pass the object from one arm to another.

In this paper we propose an implemented approach for solving the multi-arm manipulation planning problem. We illustrate our approach with a robot system consisting of three identical arms, each with six revolute degrees of freedom, operating in a 3D workspace. This system is similar to an experimental setup of three PUMA 560 arms operating in our laboratory. We restrict ourselves to the case where there is a single movable object in the environment, but we allow two types for this object: for one type, it requires two arms to move (e.g., the object is heavy); for the other type, it can be moved by a single arm at any one time. A task is specified by the description of the workspace (geometry of the obstacles, movable object, and arms), the initial and goal configurations of the movable object and arms, and the type of the movable object. The implemented planner computes a collision-free manipulation path to deliver the object to its goal configuration. It allows for the robots to change their grasp of the object, but, in its current version, it is not able to compute grasps. A set of potential grasps must be given as input, and the planner chooses among them.

Fig. 1 shows a series of snapshots along a manipulation path computed by our planner. The movable object is L-shaped and requires two arms to move.

Section 2 relates our work to previous work in manipulation planning. Section 3 gives a formal presentation of the multi-arm manipulation problem. Section 4 describes our planning approach; it introduces some simplifications to make the problem more tractable. Section 5 presents results obtained with the implemented planner.

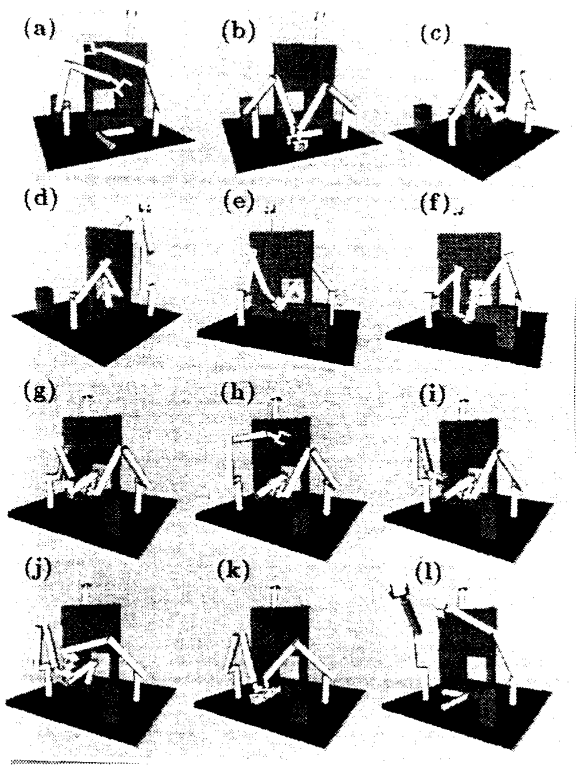

Figure 1. A multi-arm manipulation path. The object requires two arms to move it, but only one arm to hold it statically.

2 Related Work

Path planning for one robot among fixed obstacles and various extensions of this basic problem have been actively studied during the past two decades [8]. However, research strictly addressing manipulation planning is fairly recent.

The first paper to tackle this problem is by Wilfong [19]. It considers a single-body robot translating in a 2D workspace with multiple movable objects. The robot, movable objects, and obstacles are modeled as convex polygons. The robot "grasps" an object by making one of its edges coincide with an edge of the object. This definition of "grasping" extends to several movable objects. Wilfong shows that planning a manipulation path to bring the movable objects to their specified goal locations is PSPACE-hard. When there is a single movable object, he proposes a complete algorithm that runs in $O(n^3 \log^2 n)$ time, where n is the total number of vertices of all the objects in the environment. Laumond and Alami [9] propose an $O(n^4)$ algorithm to solve a similar problem where the robot and the movable object are both discs and the obstacles are polygonal.

Alami, Siméon and Laumond [1] describe a manipulation planner for one robot and several movable objects. Both the number of legal grasps of each object (positions of the robot relative to the object) and the number of legal placements of the movable objects are finite. The method was implemented for two simple robots: a translating polygon [1] and a three-revolute-joint planar arm [10]. A theoretical study of the more general case where the set of legal grasps and placements of the movable objects are manifolds (continuous sets) is presented in [10].

Our work differs from this previous research in several ways. Rather than dealing with a single robot, we consider the case of multiple cooperating robot arms moving in a 3D workspace. In addition, whereas the previous work is more theoretical in nature, our focus is more on developing an effective approach to solve manipulation planning problems of a complexity comparable to that of the tasks encountered in manufacturing (e.g., assembling, welding and/or riveting the body of a car or the fuselage of a plane) and construction work (e.g., assembling truss structures). Similarly, in [3], Ferbach and Barraquand introduce a practical approach to this manipulation planning problem using the method of variational dynamic programming.

Along a slightly different line of research, Lynch addresses the problem of planning pushing paths [13]. He establishes the conditions under which the contact between the robot and the movable object is stable, given the friction coefficients and the center of friction between the movable object and its supporting surface. These conditions yield nonholonomic constraints on the the motion of the robot.

Regrasping is a vital component in manipulation tasks. Tournassoud, Lozano-Pérez, and Mazer [18] specifically address this problem. They describe a method for planning a sequence of regrasp operations by a single arm to change an initial grasp into a goal grasp. At every regrasp, the object is temporarily placed on a horizontal table in a stable position selected by the planner. We too need to plan regrasp operations. However, the only regrasps we consider avoids contact between the object and the environment; they necessarily involve multiple arms.

The work on regrasping presented in [18] is part of an integrated manipulation system, HANDEY, described in [12]. This system controls a single PUMA arm which builds an assembly in a 3D workspace. It integrates vision, path planning, grasp planning, and motion control. While it embeds a solution to many issues not considered in this paper, it does not address the problem of planning cooperative robot motions to accomplish manipulation tasks.

Planning coordinated paths for multiple robots, without movable objects, is studied in several papers, e.g. see [14].

Grasp planning is potentially an important component of manipulation planning. In our planner, grasps are selected from a finite predefined set of grasps. A better solution for the future will be to include the automatic computation of grasps. A substantial amount of research has been done on this topic. See [16] for a commented list of bibliographical references.

We have done some prior work in manipulation planning. In [5] we propose several planners to generate manipulation paths for two identical arms in a 2D workspace. In [6] we extend one of these planners to allow the manipulation of several movable objects; this problem raises the additional difficulty of selecting the order in which the objects should be moved. Experiments have been conducted with this planner using a real dual-arm robot system developed in the Aerospace Robotics Laboratory at Stanford University [15].

Research in multi-arm manipulation planning is made more critical by the advent of new effective techniques to control cooperative robot arms (closed-loop kinematic chains). For example, see [4, 17].

3 Problem Statement

We now give a presentation of the multi-arm manipulation planning problem using a configuration space formalization. We consider only a single movable object, but for the rest, our presentation is general.

The environment is a 3D workspace $\mathcal{W}$ with p robot arms $\mathcal{A}_i$ ($i = 1, \cdots, p$), a single movable object $\mathcal{M}$, and q static obstacles $\mathcal{B}_j$ ($j = 1, \cdots, q$). The object $\mathcal{M}$ can only move by having one or several robots grasp and carry it to some destination.

Let $\mathcal{C}_i$ and $\mathcal{C}_{obj}$ be the C-spaces (configuration spaces) of the arms $\mathcal{A}_i$ and the object $\mathcal{M}$, respectively [11, 8]. Each $\mathcal{C}_i$ has dimension n_i, where n_i is the number of degrees of freedom of the robot $\mathcal{A}_i$, and $\mathcal{C}_{obj}$ is 6D. The *composite C-space* of the whole system is $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_p \times \mathcal{C}_{obj}$. A configuration in $\mathcal{C}$, called a *system configuration*, is of the form $(\boldsymbol{q}_1, \cdots, \boldsymbol{q}_p, \boldsymbol{q}_{obj})$, with $\boldsymbol{q}_i \in \mathcal{C}_i$ and $\boldsymbol{q}_{obj} \in \mathcal{C}_{obj}$. In the example of Fig. 1, $p = 3$ and $n_i = 6$, for all i.

We define the *C-obstacle region* $\mathcal{CB} \subset \mathcal{C}$ as the set of all system configurations where two or more bodies in $\{\mathcal{A}_1, \cdots, \mathcal{A}_p, \mathcal{M}, \mathcal{B}_1, \cdots, \mathcal{B}_q\}$ intersect.[1] We describe all bodies as closed subsets of $\mathcal{W}$; hence, $\mathcal{CB}$ is a closed subset of $\mathcal{C}$. The open subset $\mathcal{C} \setminus \mathcal{CB}$ is denoted by $\mathcal{C}_{free}$ and its closure by $cl(\mathcal{C}_{free})$.

For the most part we require that the arms, object, and obstacles do not contact one another. However, $\mathcal{M}$ may touch stationary arms and obstacles for the purpose of achieving static stability. $\mathcal{M}$ may also touch arms when it is being moved, in order to achieve grasp stability; then $\mathcal{M}$ can only make contact with the last link of each grasping arm (grasping may involve one or several arms). No other contacts are allowed.

This leads us to define two subsets of $cl(\mathcal{C}_{free})$:
- The *stable space* $\mathcal{C}_{stable}$ is the set of all legal configurations in $cl(\mathcal{C}_{free})$ where $\mathcal{M}$ is statically stable. $\mathcal{M}$'s stability may be achieved by contacts between $\mathcal{M}$ and the arms and/or the obstacles.
- The *grasp space* $\mathcal{C}_{grasp}$ is the set of all legal configurations in $cl(\mathcal{C}_{free})$ where one or several arms rigidly grasp $\mathcal{M}$ in such a way that they have sufficient torque to move $\mathcal{M}$. $\mathcal{C}_{grasp} \subset \mathcal{C}_{stable}$.

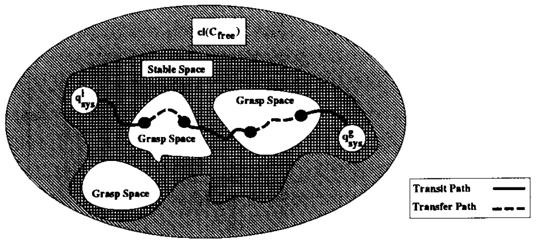

Figure 2. Components of a manipulation path and their relation to the subspaces of $cl(\mathcal{C}_{free})$.

There are two types of paths, transit and transfer paths, which are of interest in multi-arm manipulation:
- A *transit path* defines an arms' motion that does not

[1]We regard joint limits in $\mathcal{A}_i$ as obstacles that only interfere with the arms' motions.

move $\mathcal{M}$. Along such a path $\mathcal{M}$'s static stability must be achieved by contacts with obstacles and/or stationary arms. Examples of such a path involve moving an arm to a configuration where it can grasp $\mathcal{M}$ or moving an arm to change its grasp of $\mathcal{M}$. A transit path lies in the cross-section of $\mathcal{C}_{stable}$ defined by the current fixed configuration of $\mathcal{M}$.

- A *transfer path* defines an arms' motion that moves $\mathcal{M}$. It lies in the cross-section of $\mathcal{C}_{grasp}$ defined by the attachment of $\mathcal{M}$ to the last links of the grasping arms. During a transfer path, not all moving arms need grasp $\mathcal{M}$; some arms may be moving to allow the grasping arms to move without collision.

A *manipulation path* is an alternate sequence of transit and transfer paths that connects an initial system configuration, $\mathbf{q}^i_{sys}$, to a goal system configuration, $\mathbf{q}^g_{sys}$ (see Fig. 2). Some paths in this sequence may be executed concurrently as long as it does not yield collisions.

In a multi-arm manipulation planning problem, the geometry of the arms, movable object, and obstacles is given, along with the location of the obstacles. The goal is to compute a manipulation path between two input system configurations.

4 Planning Approach

We now describe our approach for solving the multi-arm manipulation planning problem. This approach embeds several simplifications, so that the corresponding planner is not fully general. Throughout our presentation, we carefully state the simplifications that we make. Some of them illustrate the deep intricacies of multi-arm manipulation planning.

Overview: A manipulation path alternates transit and transfer paths. Each path may be seen as the plan for a subtask of the total manipulation task. This yields the following two-stage planning approach: first, generate a series of subtasks to achieve the system goal configuration; second, plan a transit or transfer path for each subtask. An informal example of a series of subtasks is: grab $\mathcal{M}$, carry it to an intermediate location, change grasp, carry $\mathcal{M}$ to its goal location, ungrasp.

Unfortunately, identifying a series of subtasks that can later be completed into legal paths is a difficult problem. How can one determine whether a subtask can be completed without actually completing it? We settle for a compromise. Rather than planning for suitable transit *and* transfer subtasks, we focus solely on identifying a sequence of transfer tasks that are *guaranteed* to be completed into transfer paths. In fact, in the process of identifying these tasks, the planner also generates the corresponding transfer paths. With the transfer tasks specified, the transit tasks are immediately defined: they link the transfer paths together along with the initial and goal system configuration. Consequently, it only remains to compute the corresponding transit paths.

The assumption underlying this approach is that there exists a legal transit path for every transit task (since they are chosen without any guarantee that a transit path exists for them). Fortunately, in a 3D workspace, this is often the case. If the assumption is not verified, the planner may try to generate another series of transfer tasks, but in our current implementation it simply returns failure.

Restrictions on grasps: To simplifiy the selection of transfer tasks, we impose two main restrictions on grasps:

1. We consider two types of movable objects. An object of the first type can be moved using a single arm, for any grasp of the object. An object of the second type requires being grasped by two arms to move. The type of the movable object $\mathcal{M}$ is given as input to the planner.

2. The various possible grasps of $\mathcal{M}$ are given as a finite *grasp set*. Each grasp in this set describes a rigid attachment of the last link of an arm with $\mathcal{M}$ (first type of movable object), or a pair of such attachments (second type of movable object).

Generating transfer tasks: The generation of the transfer tasks is done by planning a path τ_{obj} of $\mathcal{M}$ from its initial to its goal configuration. During the computation of τ_{obj}, all the possible ways of grasping $\mathcal{M}$ are enumerated and the configurations of $\mathcal{M}$ requiring a regrasp are identified.

The planner computes the path τ_{obj} so that $\mathcal{M}$ avoids collision with the static obstacles $\mathcal{B}_j$. This is done using RPP (Randomized Path Planner), which is thus a component of our planner. RPP is described in detail in [2, 8].

RPP generates τ_{obj} as a list of adjacent configurations in a fine grid placed over $\mathcal{C}_{obj}$ (the 6D C-space of $\mathcal{M}$), by inserting one configuration after the other starting with the initial configuration of $\mathcal{M}$. The original RPP only checks that each inserted configuration is collision-free. To ensure that there exists a sequence of transfer paths moving $\mathcal{M}$ along τ_{obj}, we have modified RPP. The modified RPP also verifies that at each inserted configuration, $\mathcal{M}$ can be grasped using a grasp from the input grasp set. This is done in the following way. A *grasp assignment* at some configuration of $\mathcal{M}$ is a pair associating an element of the grasp set defined for $\mathcal{M}$ and the identity of the grasping robot(s). Note that the same element of the grasp set may yield different grasp assignments involving different robots. The planner enumerates all the grasp assignments at the initial configuration of $\mathcal{M}$ and keeps a list of those which can be achieved without collision between the grasping arm(s) and the obstacles, and between the two grasping arms (if $\mathcal{M}$ must be moved by two arms). We momentarily ignore the possibility that the grasping arm(s) may collide with the other arms. The list of possible grasp assignments is associated to the initial configuration. Prior to inserting any new configuration in the path being generated, RPP prunes the list of grasp assignments attached to the previous configuration by removing all those which are no longer possible at the new configuration. The remaining sublist, if not empty, is associated with this configuration and appended to the current path.

If during a down motion of RPP (a motion along the negated gradient of the potential field used by RPP) the list of grasp assignments pruned as above vanishes at all the successors of the current configuration (call it q_{obj}), the modified RPP resets the list attached to q_{obj} to contain all the possible grasp assignments at q_{obj} (as we proceed from the initial configuration). During a random motion (a motion intended to escape a local minimum of the potential), the list of grasp assignments is pruned but is constrained to never vanish. In the process of constructing τ_{obj}, the modified RPP may reset the grasp assignment list several times.

If successful, the outcome of RPP is a path τ_{obj} described as a series of configurations of $\mathcal{M}$, each anotated with a grasp assignment list. The path τ_{obj} is thus partitioned into a series of subpaths, each connecting two successive configurations. It defines as many transfer tasks as there are distinct grasp assignments associated with it. By construction, *for each such transfer task, there exists a transfer path satisfying the corresponding grasp assignment.* The number of regrasps along the generated path τ_{obj} is minimal, but RPP does not guarantee that this is the best path in that respect.

Details and comments: The condition that the same grasp assignment be possible at two neighboring configurations of $\mathcal{M}$ does not guarantee that the displacement of $\mathcal{M}$ can be done by a short (hence, collision-free) motion of the grasping arm(s). An additional test is needed when the set of grasps between two consecutive configurations is pruned. In our implementation, we assume that each arm is a non-redundant 6-DOF arm. Hence, an arm can attain a grasp with a small number of different postures, which can easily be computed using the arm's inverse kinematics. We include the posture of each involved arm in the description of a grasp assignment. Hence the same combination of arms achieving the same grasp, but with two different postures of at least one arm, defines two distinct grasp assignments. Then a configuration of $\mathcal{M}$, along with a grasp assignment, uniquely defines the configurations of the grasping arms. The resolution of the grid placed across $\mathcal{C}_{obj}$ is set fine enough to guarantee that the motion between any two neighboring configurations of $\mathcal{M}$ results in a maximal arm displacement smaller than some prespecified threshold.

A transfer path could be obstructed by the arms not currently grasping $\mathcal{M}$. Dealing with these arms can be particularly complicated. In our current implementation, we assume that each arm has a relatively non-obstructive configuration given in the problem definition (in the system shown in Fig. 1, the given non-obstructive configuration of each arm is when it stands vertical). Prior to a transfer path, all arms not involved in grasping $\mathcal{M}$ are moved to their non-obstructive configurations. The planner nevertheless checks that no collision occurs with them during the construction of τ_{obj}.

Perhaps the most blatant limitation of our approach is that it does not plan for regrasps at configurations of $\mathcal{M}$ where it makes contact with obstacles (as we said, τ_{obj} is computed free of collisions with obstacles). Since the object cannot levitate, we require that $\mathcal{M}$ be held at all times during regrasp. We assume that if $\mathcal{M}$ requires more than one robot to move, any subset of a grasp is sufficient to achieve static stability during the regrasp. For example, if a grasp requires two robots, anyone of these robots, alone, achieves static stability, allowing the other robot to move along a transit path. An obvious example where this limitation may prevent our planner from finding a path is when the robot system contains a single arm; no regrasp is then possible.

RPP is only probabilistically complete [2]. If a path exists for $\mathcal{M}$, it will find it, but the computation time cannot be bounded in advance. Furthermore, if no path exists, RPP may run for ever. Nevertheless, for a rigid object (as is the case for $\mathcal{M}$), RPP is usually very quick to return a path, when one exists. Hence, we can easily set a time limit beyond which it is safe to assume that no path exists. Other path planners could possibly be used in place of RPP.

Generation of transit paths: The transfer tasks identified as above can be organized into successive layers, as illustrated in Fig. 3. Each layer contains all the transfer tasks generated for the same subpath of τ_{obj}; the transfer tasks differ by the grasp assignment. Selecting one such task in every layer yields a series of transit tasks: the first consists of achieving the first grasp from the initial system configuration; it is followed by a possibly empty series of transit tasks to change grasps between two consecutive transfer tasks; the last transit task is to achieve the goal system configuration. Hence, it remains to identify a grasp assignment in each layer of the graph shown in Fig. 3, such that there exist transit paths accomplishing the corresponding transit tasks.

Assume without loss of generality that all arms are initially at their non-obstructive configurations. Our planner first chooses a transfer task (anyone) in the first layer. Consider the transit task of going from the initial system configuration to the configuration where the arms achieve the grasp assignment specified in the chosen transfer task, with $\mathcal{M}$ being at its initial configuration. The coordinated path of the arms is generated using RPP. If this fails, a new attempt is made with another transfer task in the first layer; otherwise, a transfer task is selected in the second layer. The connection of the system configuration at the end of the first transfer task to the system configuration at the start of this second transfer task forms a new transit task.

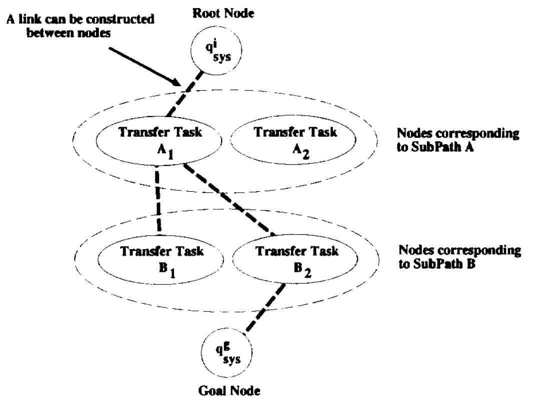

Figure 3. The directed and layered graph.

The transit task between two transfer tasks is more difficult to solve. To understand the difficulty, imagine the case where $\mathcal{M}$ is a long bar requiring two arms to move. Assume that the robot system contains only two arms and that the bar can be grasped at its two ends and at its center. Consider the situation where the bar is grasped at its two ends and the regrasp requires swapping the two arms. This regrasp is not possible without introducing an intermediate grasp. For example: arm 1 will ungrasp one end of the bar and regrasp it at its center (during this regrasp, arm 2 will be holding the bar without moving); then arm 2 will ungrasp the bar and regrasp it at the other end; finally, arm 1 will ungrasp the center of the bar and will regrasp it at its free end. Fig. 4 illustrates this example.

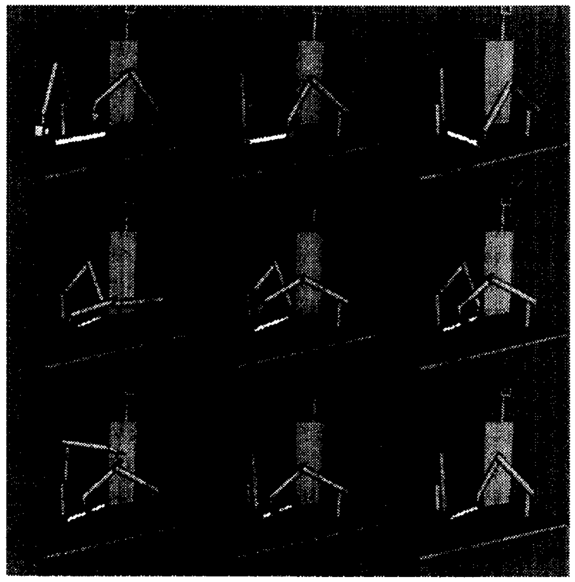

Figure 4. An example illustrating the complexity of changing grasps.

We address this difficulty by breaking the transit task between two transfer tasks into smaller transit subtasks. Each transit subtask consists of going from one grasp assignment to another in such a way that no two arms use the same grasp at the same time. In this process, we allow arms not involved in the first and last assignment to be used. We start with the first grasp assignment and we generate all the potential grasp assignments that we may be able to achieve from it (assuming the corresponding transit paths exist). We generate the successors of these new assignments, and so on until we reach the desired assignment (the one used in the next transfer task). For each sequence that achieves this desired assignment, we test that it is actually feasible by using RPP to generate a transit path between every two successive grasp assignments. We stop as soon as we obtain a feasible sequence. The concatenation of the corresponding sequence of transit paths forms the transit path connecting the two considered transfer tasks. We then proceed to link to the next layer of transfer tasks.

When we reach a transfer path in the last layer, its connection to the goal system configuration is carried out in the same way as the connection of the initial system configuration to the first layer.

5 Examples of Generated Paths

We implemented the above approach in a planner written in C and running of a DEC Alpha workstation under UNIX. All experiments so far were conducted with a robot system made of 3 identical arms, each with 6 revolute joints. The non-obstructive configuration of each arm is one where the arm stands vertical.

Fig. 1 shows a manipulation path generated by the planner for an L-shaped object. This object requires two arms to move it. The object is taken through the window in the large obstacle located in the middle of the workspace. Notice that in the change of grasp, at least one arm is holding the object at all times. For this path, it took 45 seconds to identify the transfer tasks and 30 additional seconds to complete the manipulation path. For the generation of τ_{obj}, the object's C-space was discretized into a $100 \times 100 \times 100$ grid. For the generation of the transit paths, the joint angles of the arms were discretized into intervals of 0.05 radians. The grasp set contains 64 grasps, yielding grasp assignment lists up to around 15,000 elements.

Fig. 4 shows a manipulation path generated for a long bar requiring two arms to move it. This example illustrates the complexity of changing grasps. For this path, it took 25 seconds to identify the transfer tasks and an additional 20 seconds to complete the manipulation path. The same discretizations as above were used. The grasp set of the long bar contains 24 grasps, yielding grasp assignment lists up to around 3,000 elements.

Fig. 5 shows a manipulation path found for a T-shaped object. This object requires a single arm to move it, and only two arms are used along the computed path. One arm first grasps the object at one end of the T. It passes the object to another arm that grasps it at its other end and brings it to its goal configuration. For this path, the planner took 40 seconds to identify the transfer tasks and then another 25 seconds to complete the manipulation path. The same discretizations as above were used. The grasp set of the T-shaped object contains 49 grasps. It yields grasp assignment lists up to around 2,000 elements.

For these examples, in computing the transit paths RPP uses the sum of the angular joint distances to the goal configuration as the guiding potential. However, in computing τ_{obj} RPP uses an NF2-based potential with three control points [8]. For both cases of finding the transit paths and τ_{obj}, we limit the amount of computation spent in RPP to three backtrack operations [8], after which the planner returns failure. Failure to find τ_{obj} results in the immediate failure to find a manipulation path. Similarly, a failure to find transit paths to link together the layers of transfer paths results in a failure to find a manipulation path. The time for the planner to report its failure depends on the problem, with some examples being from 30 seconds to a few minutes.

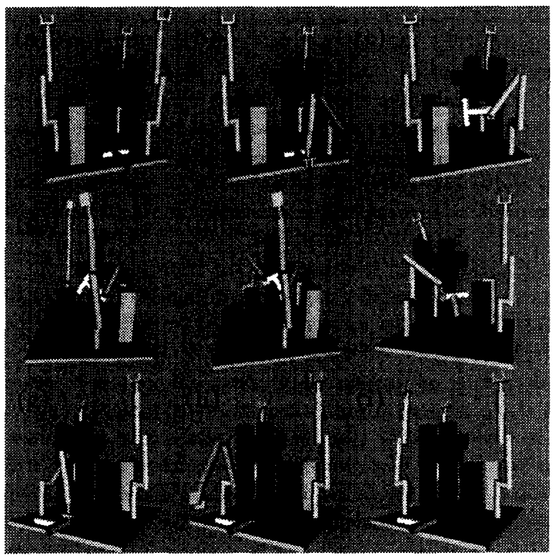

Figure 5. Another example of a manipulation path. The object is T-shaped and requires only one arm to manipulate it.

6 Conclusion

Multi-arm manipulation planning is a new motion planning problem with application in various tasks, such as assembly and construction. We have presented an approach for solving this complicated problem. Our approach embeds several simplifications yielding an implemented planner that is not fully general. However, experiments with this planner show that it is quite reliable and fast in finding manipulation paths, when such paths exist, making it suitable as an interactive tool to facilitate robot programming. We believe the robust nature of the planner is the result of careful consideration of the general manipulation problem, the introduction of reasonable simplifications, and the appropriate utilization of the efficient randomized path planner.

Most of our experiments so far have been conducted with simulated mechanical arms. However, we have completed a first integration of the planner with a software module simulating human arms which incorporates the inverse kinematics of such arms [7]. The result is a system that computes natural human-arm manipulation motions. The goal of this work is to help in the interactive generation of animated video sequences for studying task ergonomy. Also, such videos would usefully replace cumbersome instruction manuals accompanying some assembly kits.

Our next step is to connect our planner to the controller of our 3-PUMA system and experiment with the paths generated by the planner to verify our assumptions in the real world. Later, we hope to include the ability to regrasp the movable object by having the arms place it in a stable configuration against some obstacles. We also plan to use existing results to automatically compute the grasp set of an object from its geometric model. The technique described in [6] to deal with multiple movable objects in a 2D space should also be applicable in our planner. Furthermore, like in [6], we plan to allow parallel execution of consecutive transfer and transit paths as long as this parallelism does not yield collisions. Finally, we hope to incorporate considerations of torque constraints and dynamics issues in the planning process. Indeed, for the regrasping actions we currently assume that a single arm is sufficient to hold the object statically, when actually there may be certain configuations where the actuator torques are too limited. Furthermore, allowing the object to slip in the grasp of the robots due to gravitational or inertial effects may be another way of changing grasps.

References

[1] R. Alami, T. Siméon and J.P. Laumond, "A Geometrical Approach to Planning Manipulation Tasks: The Case of Discrete Placements and Grasps," in *Robotics Research 5*, H. Miura and S. Arimoto, eds., MIT Press, Cambridge, 1990, pp. 453-459.

[2] J. Barraquand and J.C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," *Int. J. Robotics Research*, 10(6), December 1991.

[3] P. Ferbach and J. Barraquand, *A Penalty Function Method for Constrained Motion Planning*, Rep. No. 34, Paris Research Lab., DEC, Sept. 1993.

[4] O. Khatib, "Object Manipulation in a Multi-Effector Robot System," *Robotics Research 4*, R. Bolles and B. Roth, eds., MIT Press, Cambridge, MA, 1988, pp. 137-144.

[5] Y. Koga and J.C. Latombe, "Experiments in Dual-Arm Manipulation Planning," *Proc. IEEE Int. Conf. Robotics and Automation*, Nice, France, 1992, pp. 2238-2245.

[6] Y. Koga, T. Lastennet, J.C. Latombe, and T.Y. Li "Multi-Arm Manipulation Planning," *Proc. 9th Int. Symp. Automation and Robotics in Construction*, Tokyo, June 1992.

[7] K. Kondo, *Inverse Kinematics of a Human Arm*, in preparation, 1993.

[8] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[9] J.P. Laumond and R. Alami, *A Geometrical Approach to Planning Manipulation Tasks: The Case of a Circular Robot and a Movable Circular Object Amidst Polygonal Obstacles*, Rep. No. 88314, LAAS, Toulouse, 1989.

[10] J.P. Laumond and R. Alami, *A Geometrical Approach to Planning Manipulation Tasks in Robotics*, Rep. No. 89261, LAAS, Toulouse, 1989.

[11] T. Lozano-Pérez, "Spatial Planning: A Configuration Space Approach," *IEEE Tr. Computers*, 32(2), 1983, pp. 108-120.

[12] T. Lozano-Pérez et al., "Handey: A Task-Level Robot System," *Robotics Research 4*, R. Bolles and B. Roth, eds., MIT Press, Cambridge, MA, 1988, pp. 29-36.

[13] K.M. Lynch, "Planning Pushing Paths," *Proc. JSME Int. Conf. Advanced Mechatronics*, Tokyo, 1993, pp. 451-456.

[14] P.A. O'Donnell and T. Lozano-Pérez, "Deadlock-Free and Collision-Free Coordination of Two Robot Manipulators," *Proc. IEEE Int. Conf. Robotics and Automation*, 1989, pp. 484-489.

[15] G. Pardo-Castellote, T.Y. Li, Y. Koga, R.H. Cannon, J.C. Latombe, and S.A. Schneider, "Experimental Integration of Planning in a Distributed Control System," *Preprints 3rd Int. Symp. Experimental Robotics*, Kyoto, October 1993.

[16] J. Pertin-Troccaz, "Grasping: A State of the Art," in *The Robotics Review 1*, O. Khatib, J.J. Craig, and T. Lozano-Pérez, eds., MIT Press, Cambridge, MA, 1989, pp. 71-98.

[17] S.A. Schneider and R.H. Cannon, "Object Impedance Control for Cooperative Manipulation: Theory and Experimental Results," *IEEE Tr. Robotics and Automation*, 8(3), 1992, pp. 383-394.

[18] P. Tournassoud, T. Lozano-Pérez, and E. Mazer, "Regrasping," *Proc. IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, 1987, pp. 1924-1928.

[19] G. Wilfong, "Motion Planning in the Presence of Movable Obstacles," *Proc. 4th ACM Symp. Computational Geometry*, 1988, pp. 279-288.