The 9th International Symposium on Automation and Robotics in Construction June 3-5, 1992 Tokyo, Japan

Multi-Arm Manipulation Planning

Yoshihito Koga, Thibaud Lastennet, Jean-Claude Latombe, and Tsai-Yen Li
Robotics Laboratory
Department of Computer Science, Stanford University
Stanford, CA 94305, USA

ABSTRACT

Construction requires manipulating objects of various sizes and weights, including long and heavy pipes and beams. To address this need we investigate automatic multi-arm manipulation from two complementary perspectives: control and planning. This paper, which only presents one aspect of our research, focuses on the planning issues underlying multi-arm manipulation. It describes an implemented software system that computes collision-free paths for two arms in order to transport objects from their initial locations (positions and orientations) to specified goal locations. This system demonstrates several kinds of interactions between the two arms: cooperative manipulation of long objects, independent manipulation of small objects, and transfer of objects from one arm to the other to increase the workspace size.

ACKNOWLEDGMENTS: This research was partially funded by CIFE (Center for Integrated Facility Engineering) and DARPA contract DAAA21-89-C0002 (Army). Y. Koga was supported in part by a Canadian NSERC Fellowship.

1. INTRODUCTION

Material handling plays a key role in construction. Many operations require manipulating objects of various sizes and weights, e.g. long and heavy pipes and beams. Such operations are often slow and dangerous. Several attempts have been made to partially automate them [10]. For example, teleoperated manipulators are used to lift, erect and assemble pipes [1, 3, 4]. However, they still are not very time-efficient. In our opinion, there are several reasons for that. One, at the control level, is that long and heavy objects put high constraints on poorly controlled actuators. Another, at the planning level, is that complex paths have often to be generated on the fly to avoid various existing obstacles.

To address this need, as well as similar needs in other domains (e.g., assembly in space), we are conducting research to investigate multi-arm manipulation from two complementary perspectives, control and planning [5]. (By "arm", we mean any kind of machine capable of moving objects; for example, a crane is seen here as one particular type of arm [9].) The purpose of using several arms is to ultimately make manipulation more reliable, faster, and more cost-effective. Indeed, in many occasions, single-arm manipulation is not sufficient. For example, heavy or long objects put excessive torque constraints on actuators, and space accessibility is too limited. On the other hand, although promising, multi-arm systems require decisive technical advances in control and planning for cooperative manipulation.

This paper focuses on the planning issues underlying multi-arm manipulation. (See [11] for a description of the work on control issues.) It describes an implemented software system that computes the paths of two arms to transport objects from their initial locations (positions and orientations) to specified goal locations among obstacles. This system takes CAD-type models as input (these models could also be obtained through visual sensing). It demonstrates several kinds of interactions between the two arms:

- The two arms move cooperatively to transport the same object. This interaction is particularly useful to deal with long and/or heavy objects that an arm could not carry alone.
- One arm hands one object over to the other arm that carries the object further away. This interaction allows the two arms to combine their accessible workspace into a larger one.
- The two arms move simultaneously to carry different objects. When feasible (i.e., if no collision is at risk), this parallel execution of independent motions reduces time delays.

Our software integrates several efficient planning methods developed in our laboratory over the past few years [7, 2, 6], as well as some new techniques. It is written in C and runs under UNIX on a DEC 5000 workstation. It is equipped with a graphic simulator to visualize the generated motion plans. It has also been partially experimented with real robots in the Aerospace Robotics Laboratory at Stanford University. In its current version, the system operates in a two-dimensional workspace. However, we have developed path planners for three-dimensional workspaces [2], and we plan to extend our manipulation planner to three-dimensional workspaces as well. In any case, the functions provided by our system are new, and no previous implemented motion planner has demonstrated such a combination of capabilities.

2. Scenario and Example

Throughout this paper we consider the following scenario illustrated in Fig. 1:

The workspace is a rectangular area containing fixed obstacles shown black in the figure. Two identical 3-revolute-joint planar arms are available, which are shown in their resting configurations at the bottom-left corner and the top-right corner of the workspace. The revolute joints are displayed as small squares (fixed ends of the arms) and disks (other joints); the third joint of each arm corresponds to a rotating wrist. We will call ARM_1 the arm at the bottom-left corner, and ARM_2 the arm at the top-right corner. Three *movable objects* (shown grey) are delivered on the left-hand side of the workspace in three distinct feeders. We will call each such object, OBJ_1, OBJ_2, or OBJ_3, according to whether it is delivered by the top, central, or bottom feeder, respectively.

The two arms operate in a horizontal plane located above the obstacles and the movable objects. They grasp movable objects by positioning their wrists at points marked + (and activating a mechanical, magnetic or suction device). Hence, the only possible collisions are between a movable object and an obstacle, or between two movable objects, or between the two arms. We assume that no mechanical joint limits restrict the motions of the arms in the rectangular workspace.

The geometric models of the obstacles and the movable objects are described in a file that can be easily modified. For each type of movable object, this file specifies the number of arms (1 or 2) that are needed to carry the object and the location of the wrist to grasp them (position of the + marks). In the setting of Fig. 1, OBJ_1 and OBJ_3 (top and bottom feeders) can be moved by a single arm, while the long object provided by the central feeder (OBJ_2) requires two arms. There are two possible grasps of OBJ_2 obtained by swapping the grasping positions of the two arms.

The above setting is shown on the user's graphic display. Using the mouse, the user clicks on the objects fed on the left, changes their orientations, and drags them to anywhere in the workspace, in order to specify their goal positions and orientations. Fig. 2 displays an example of goal locations selected by the user for the three movable objects.

The implemented system performs some very quick preliminary tests to avoid the user asking for an impossible goal. For example, it checks that the goal location of every movable object is collision-free. It also checks that a grasp of each movable object, at its goal location, is attainable by the arms.

The problem for the planner is to find a sequence of collision-free paths for the arms whose execution bring the movable objects from their initial locations in the feeders to the goal locations specified by the user. We call this sequence of paths a *manipulation path.* Our planner is fast enough to generate manipulation paths on-line. The execution of the generated plans is immediately simulated on the graphic display. Fig. 3 and 4 show a series of twelve snapshots along a manipulation path generated by our planner for the goal locations shown in Fig. 2. The generated manipulation sequence (the only valid one in this example) is 3-1-2. The arm ARM_1 first grasps OBJ_3 and moves it toward its goal position (snapshots b and c). It cannot reach it and, hence, hands the object to ARM_2 that brings it to the goal while ARM_1 returns to its resting configuration (snapshot d). From there ARM_2 moves toward OBJ_1 (snapshot e), grasps it, and moves it to its goal (snapshots f and g). Then both arms move to grasp OBJ_2 (snapshot h). They carry the OBJ_2 toward its goal, but on the way they have to swap grasps to avoid collision between them (snapshots i and j). With the new grasp, they attain the goal of OBJ_2 (snapshots k and l) from where they return to their resting configurations (not shown). The total computing time for this path was about 17 seconds.

3. Overview of the Planner

Our planner generates a manipulation path according to the following three steps:

1. MANIPULATION SEQUENCING: The planner first determines the order in which the arms will carry the movable objects to their goal locations. For example, in Fig. 2, moving OBJ_2 to its goal location before OBJ_1 would make it impossible to later bring OBJ_1 to its goal location; hence, this must be avoided.
2. ARM PLANNING: The planner then computes a collision-free path of the arms to carry each movable object from the feeder to its goal location. If the movable object is OBJ_2, the two arms move cooperatively. If the object is OBJ_1 or OBJ_3, one arm moves at a time in the generated path.
3. MOTION PARALLELIZATION: Three paths are now available, one for every movable object. They can safely be executed in sequence; but this may not be very efficient. In this last step, the planner determines the extent to which the paths can be executed in parallel without causing collision.

These three steps are described in more detail in the following sections.

4. Manipulation Sequencing

Manipulation sequencing is done by planning the paths of the movable objects as if they were able to move by themselves (that is, without the arms). The planner enumerates every possible sequence of manipulation, checks that it is feasible relative to the only constraints arising from the movable objects and the obstacles, and returns the first valid sequence found.

Since, in general, the movable objects constrain each other much more at their goal locations than at their initial locations where they are well separated, the planner checks each sequence backward, so that an invalid sequence is disqualified quicker. For example, assume that the sequence to be checked is 1-2-3. The sequencing algorithm proceeds as follows. It first considers the last object in the sequence, i.e. OBJ_3, and attempts to generate a collision-free path to move it from its initial to its goal location, assuming that both OBJ_1 and OBJ_2 are at their goal locations. If it succeeds, it then attempts to generate a path for OBJ_2, assuming that OBJ_1 (but not OBJ_3) is at its goal location. If it succeeds again, it finally attempts to find a path for OBJ_1 with both OBJ_2 and OBJ_3 at their initial locations. If this last attempt succeeds, the sequence 1-2-3 is returned.

If the sequencing algorithm fails to generate a path for an object in a sequence, it either considers another sequence, or it exits with failure. For example, if it fails to find a path for OBJ_3 while OBJ_1 and OBJ_2 are at their goal locations, it permutes OBJ_2 and OBJ_3, and attempts to generate a path for OBJ_2, with OBJ_1 and OBJ_3 at their goal locations. If this is necessary, it eventually considers all possible permutations of the list of objects. It fails definitely when all permutations have been unsuccessfully considered.

Once a sequence is generated (if one is generated), it is passed to the next module of the planner, which plans paths for the two arms and the movable objects according to this sequence (see Section 5). However, the fact that a sequence is feasible for just the movable objects (that is, ignoring the two arms) is only a necessary condition for the existence of paths when the two arms are considered. Thus, it may happen that the second planning module fails. Then the sequencing algorithm is invoked again and generates a new permutation, and so on. The iteration ends when the sequencing algorithm cannot generate any new permutation.

If there were n movable object, the planner could have to check up to $n!$ permutations. However, a single failure may disqualify several permutations simultaneously. Hence, many permutations do not have to be considered in most cases. For example, in our three-object scenario, assume that the sequence 1-2-3 is being checked. If no path can be found for OBJ_3 (both OBJ_1 and OBJ_2 are at their goal locations), both the sequences 1-2-3 and 2-1-3 are eliminated. If no path can be found for OBJ_2 (OBJ_1 is at its goal location and OBJ_3 at its initial location), both the sequences 1-2-3 and 1-3-2 are discarded. If no path can be found for OBJ_1 (both OBJ_2 and OBJ_3 are at their initial locations), all sequences are disqualified. And so on. However, the worst-case computational complexity remains proportional to $n!$.

To compute the paths of the movable objects, the sequencing algorithm uses the potential-field planner described in [2], with the improvements described in [8]. This planner explores a three-dimensional grid placed over the configuration space of the moving object in a best-first fashion using a potential function U as the cost function. The function U is itself computed by combining two local-minima-free potential functions precomputed over the two-dimensional workspace. (Various techniques to perform these computations are described in [7].) The resulting path planner is extremely fast. Its implementation in C on a DEC 5100 workstation typically generates the path of an object in a few tenths of a second.

5. Arm Planning

Once a manipulation sequence has been established, the planner computes collision-free paths of the arms to transfer the movable objects from their initial locations to their goal locations in that sequence. The planner essentially reuses the potential-field planning method mentioned above, with some additional checks (object graspability) as described below (see [6] for more detail).

The planner recomputes a path of each movable object with the other two movable objects located according to the established sequence of manipulation. For example, if the sequence is 1-2-3, the path of OBJ_2 is computed with OBJ_1 at its goal location and OBJ_3 at its initial location. During this computation, the planner uses the simple inverse kinematics of the arms to check that the movable object is graspable by the arms at every location along the constructed path with no collision between the two arms; a non-graspable location of the object is now considered invalid, just as a non-collision-free one. A location of OBJ_1 or OBJ_3 is a graspable one if the grasp point of the object is attainable by at least one arm. A location of OBJ_2 is graspable if one grasp point is attainable by one arm and the other grasp point by the other arm. Notice that we could have reused the paths of the movable objects constructed to determine the manipulation sequence; but these paths may not satisfy the graspability condition and may have to be changed. Since it is so fast to recompute them, we rather chose to re-start the computation from scratch and have a more modular software.

When the path of an object has been computed, together with the potential grasps by the arms along this path, the planner selects among these grasps those which will be used. Starting at the initial location of the object, it selects the grasp that remains valid along the longest consecutive portion of the path. Just before this grasp becomes invalid (if this occurs), it selects another grasp among the potential ones (again the one which will remain valid the longest). And so on, until the goal location of the movable object is attained. Using simple path planning techniques and the arms' inverse kinematic equations, the planner then generates the paths of the arms to attain the selected grasps and to move the object with these grasps.

This simple general method allows us to handle the three types of arm interactions listed in the introduction and illustrated in Fig. 3 and 4. In the case of OBJ_1 and OBJ_3, changing grasp consists of passing the object from one arm to the other. In the case of OBJ_2, it consists of swapping the two arms.

6. Motion Parallelization

At this point, the planner has constructed a series of collision-free paths for the arms (manipulation path). Whenever a movable object reaches its goal position, the arm(s) carrying it return to their resting configurations at the corners of the workspace. Furthermore, when OBJ_1 or OBJ_3 is moved, only one arm moves at a time. This sequential manipulation may cause a lot of time to be wasted.

The planner now attempts to simplify and parallelize the arms' motions as follows. It starts at the end of the manipulation path and moves backward toward the beginning of this path. Whenever it encounters two consecutive paths to be executed by the same arm, it simplifies the path so that the arm does not return to its resting configuration in the corner of the workspace between the two motions; it computes a path of the arm going directly to one grasp position to the next. If, instead, two consecutive paths have to be executed by two different arms, the planner determines the extent to which the two paths can be executed concurrently. Assume that τ_1 and τ_2 are two consecutive paths, with τ_1 before τ_2, τ_1 being a path of one arm alone, and τ_2 being a path of the other arm alone. Each of the two paths are described as a fine sequence of discrete configurations of the moving arm (plus the carried movable object, if any). The planner tries to start τ_2 during τ_1 and checks whether the two arms (and, possibly, the carried movable objects) collide while their are moving simultaneously. In order to obtain the best overlap between the two paths, the planner proceeds dichotomically: it first attempts to start τ_2 at the middle of τ_1; if this parallelization causes no collision, it attempts to start τ_2 at the first quarter of τ_1, otherwise it attempts to start τ_2 at the third quarter of τ_1; and so on. Collision checking is done only between the two arms and, if each arm carries a movable object, between OBJ_1 and OBJ_3. Collision checking is done only at the discretized configurations described in the paths.

Notice that the result of this last step is in general not optimal. First, the manipulation sequence that was selected in the first step may not allow maximal parallelization. Second, except when two consecutive paths are executed by the same arm, the above computation does not change the geometry of the paths; it only make them overlap in time. Between most moves, each arm returns to its resting configuration. Further optimization is possible, but at a much greater computational cost.

7. Conclusion

We have described a planner for a two-arm robotic system manipulating objects in a workspace cluttered by obstacles. This planner automatically generates collision-free paths of the two arms to carry the objects from their initial locations to the specified goal locations. The planner allows several kinds of interactions between the two arms. This research is part of a larger project aimed at investigating material handling in construction by multi-arm systems. We ultimately expect a planner similar to the one we have developed to be part of robot controllers in order to allow human operators to have high-level supervision over arms' motions (semi-autonomous robots) without being bugged by the details of the motions.

Our planner has still some important limitations. (1) The workspace is two-dimensional, while most construction tasks occur in a three-dimensional environment; we have developed path planners that can operate in three-dimensional workspaces [2], but they are computationally more demanding. (2) Our planner is unable to solve manipulation problems that require interweaving the motions of the various movable objects by transporting some of them to intermediate locations; currently, each object is directly moved to its goal location. (3) At their goal locations, the movable objects most have no contact between them or with the workspace obstacles; recent work in assembly planning [12] will soon be incorporated in our planner to eliminate this limitation. (4) Finally, we currently work on taking dynamic constraints into account in our planner and connecting it to real robots.

References

[1] D. Alciatore, **Automation of a Piping Construction Manipulator and development of a Heuristic Application-Specific Path Planner**, Ph.D. Dissertation, University of Texas, Austin, TX, December 1989.

[2] J. Barraquand and J.C. Latombe, "Robot Motion Planning: A Distributed Representation Approach," *The Int. J. of Robotics Research*, MIT Press, Cambridge, MA, 10(6), 628–649, December 1991.

[3] D. Fischer, **Piping Erection Constructability Issues in a Semi-Automated Environment**, Ph.D. Dissertation, University of Texas, Austin, TX, May 1989.

[4] C.C. Glass, **The Pipe Manipulator: A Complete Assessment of a New Idea in Construction Technology**, Thesis, The University of Texas, Austin, December 1984.

[5] O. Khatib and J.C. Latombe, **Cooperative Manipulation of Pipes in Construction Tasks**, Research Proposal, Center for Integrated Facility Engineering (CIFE), Stanford University, Stanford, CA, 1990.

[6] Y. Koga and J.C. Latombe, "Experiments in Dual-Arm Manipulation Planning," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Nice, France, May 1992.

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

[8] J.C. Latombe, "A Fast Path Planner for a Car-Like Indoor Mobile Robot," *Proc. of the 9th Nat. Conf. on Artificial Intelligence*, Anaheim, CA, 659–665, July 1991.

[9] Y. Rosenfeld and S. Berkovitz, "Automation of Existing Tower Cranes, Economic and Safety Examination," *Proc. of the 6th Int. Symp. on Automation and Robotics in Construction*, San Francisco, CA, 1989.

[10] Y. Sagawa and Nakahara, "Robots for the Japanese Construction Industry," *SABSE Proc. P-86.85, SABSE Periodica*, 2, 82-91

[11] D. Williams and O. Khatib, "Generalized Compliance and Constrained Motion in Multi-Arm Manipulation," *proc. of the 2nd Int. Symp. on Measurement and Control in Robotics*, Tsukuba, Japan, November 1992

[12] R.H. Wilson, **On Geometric Assembly Planning**, Ph.D. Dissertation, Dept. of Computer Science, Stanford University, Stanford, CA, 1992.

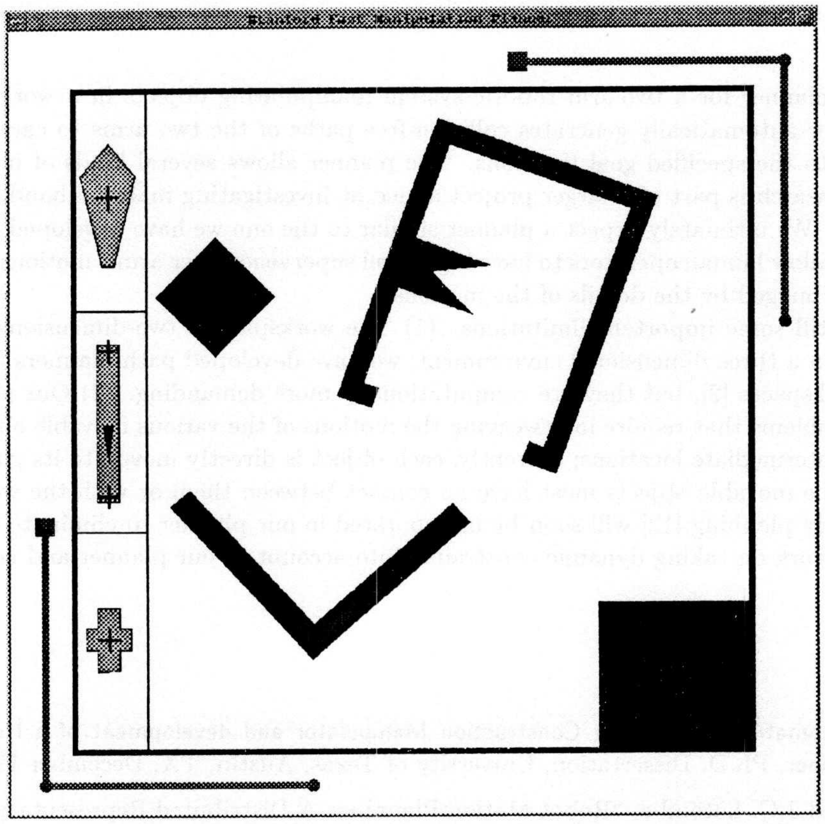

Figure 1: Dual-Arm Manipulation Setting

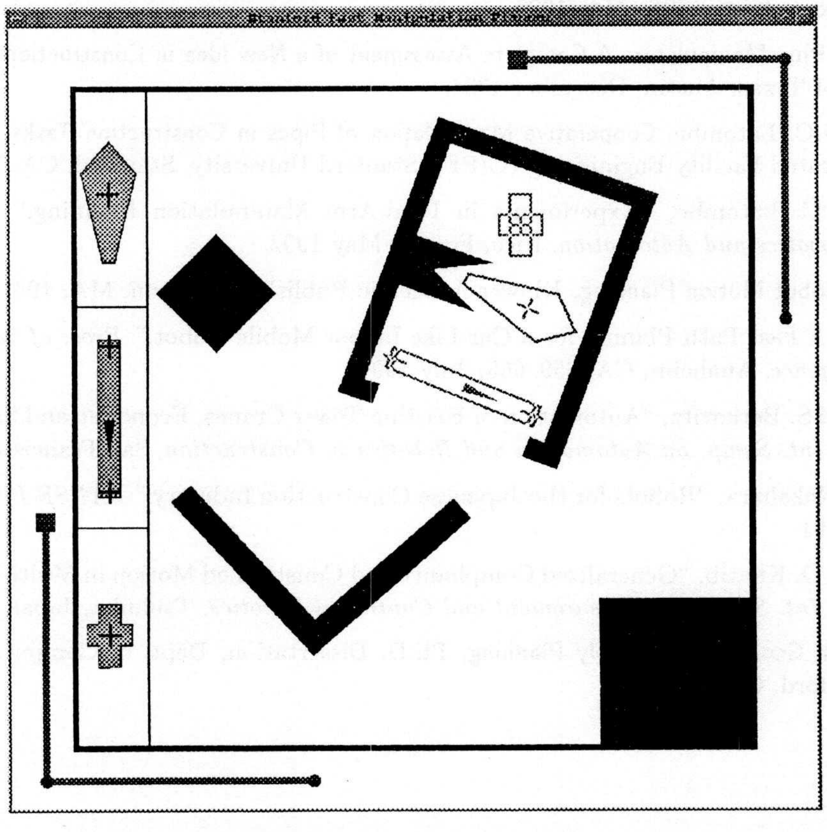

Figure 2: Goals of Movable Objects

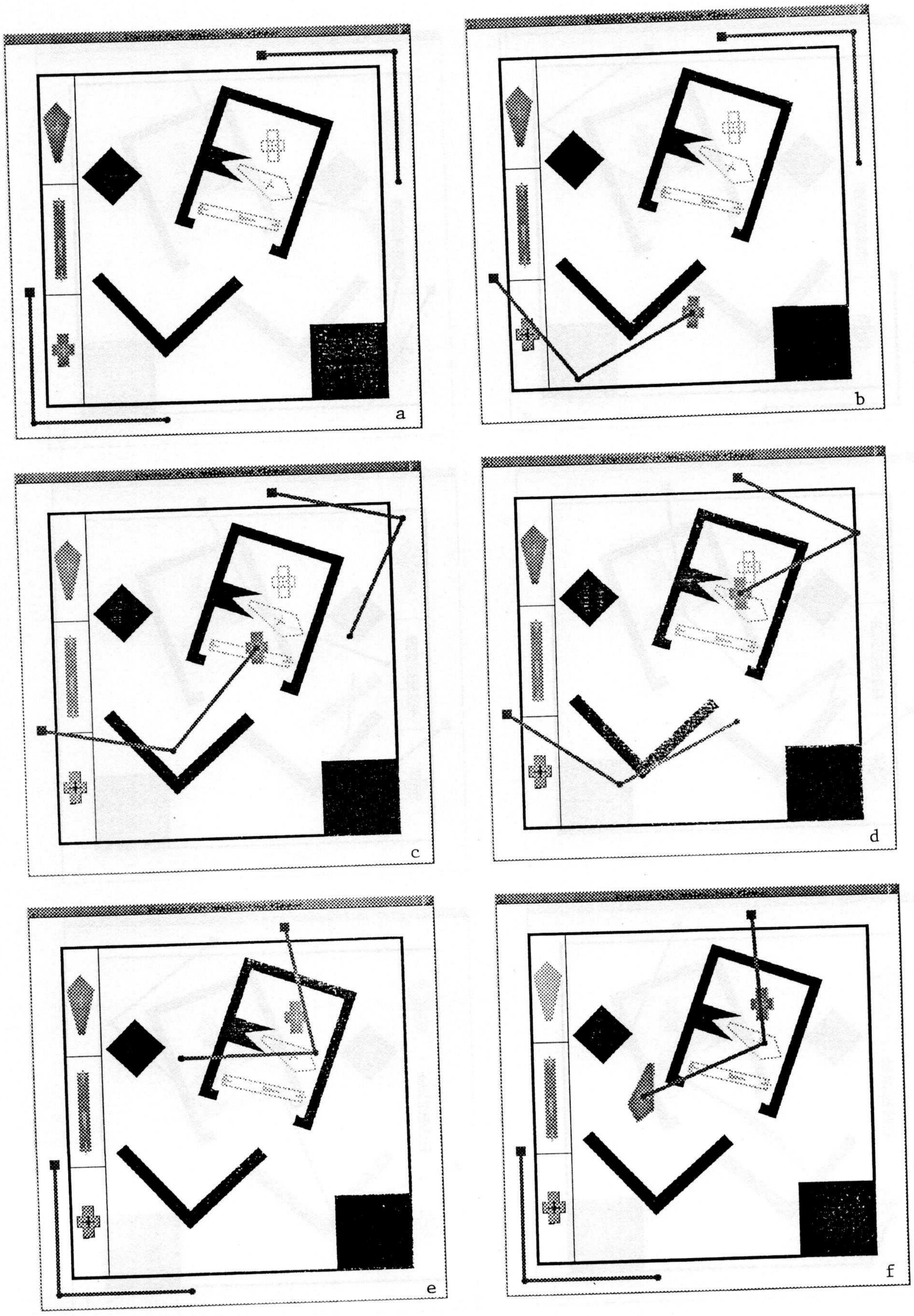

Figure 3: Manipulation Path (beginning)

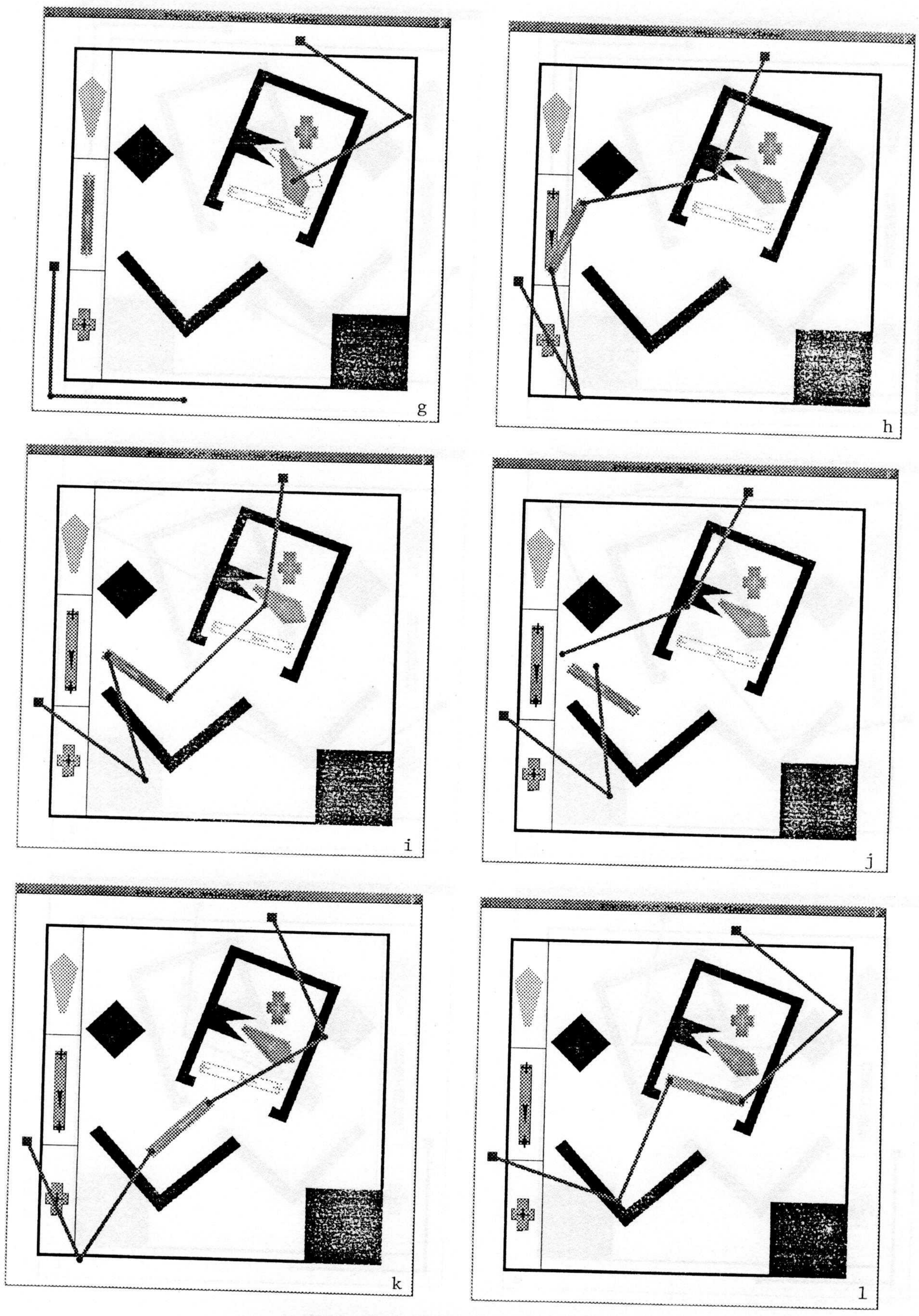

Figure 4: Manipulation Path (end)