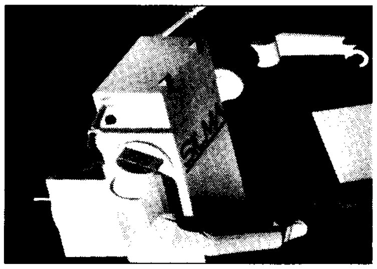


Task-Level Planning of Pick-and-Place Robot Motions

Tomás Lozano-Pérez, Joseph L. Jones, Emmanuel Mazer, and Patrick A. O'Donnell
MIT Artificial Intelligence Laboratory

A task-level robot system is one that can be instructed in terms of task-level goals, such as "Grasp part *A* and place it inside box *B*." This type of specification contrasts sharply with that required for existing industrial robot systems, which insist on a complete specification of each motion of the robot and not simply a description of a desired goal. An important characteristic of task-level specifications is that they are independent of the robot performing the task, whereas a motion specification is wedded to a specific robot.

Task-level robot systems have long been a goal of robotics research. As early as 1961, Ernst's PhD thesis at MIT[1] attempted to develop such a system. Since then, a variety of task-level robot systems have been proposed, and several of them have seen some level of implementation. Lozano-Pérez and Taylor provide a survey of previous work in this area.[2]

For the past three years, we have been developing a task-level robot system named Handey. The current system is by no means a complete task-level system; it is limited to *pick-and-place* operations, that is, picking up a part and placing it at a specified destination. The current implementation has successfully carried out dozens of pick-and-place operations involving a variety of parts in relatively complex environments. Figure 1 shows a sequence of intermediate steps of a pick-and-place operation planned by Handey from a specification of the desired final position of the part and geometric and kinematic models of the robot and the environment. The steps illustrated in the figure are

(1) the initial position of the robot and the parts,
(2) grasping a part at a location chosen by the system to avoid collisions with nearby objects,
(3) placing the part on the table,
(4) regrasping the part at a new location compatible with the environment at the destination,
(5) placing the part at the specified destination, and returning to the initial position.

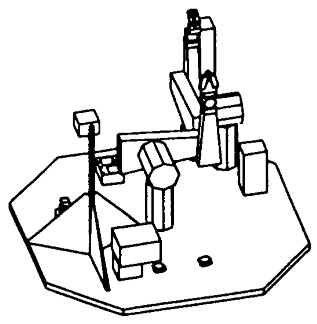
The Handey system breaks the pick-and-place problem into nearly independent, computationally feasible subproblems as a step toward a comprehensive task-level system.

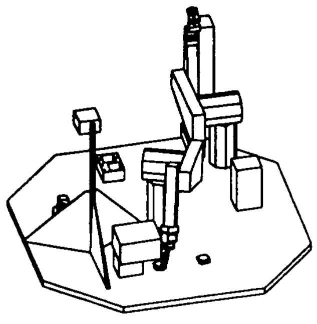
The pick-and-place problem

Consider the overall problem faced by a task-level robot system, given a mechanical assembly task and a single, sufficiently precise robot. (In this article, we assume that the position of all the parts are known to high accuracy and the robot's positioning accuracy is sufficient to carry out the assembly simply by moving the part to a fixed target position. If this assumption is not satisfied, then assembly planning must include *fine-motion* planning, that is, the synthesis of motion strategies that can achieve a desired goal in the presence of sensing and control error.[3,4]) We can summarize the overall assembly planning problem, of which pick-and-place is but a part, as follows:

- Choose an order of assembly for the parts.
- Choose initial positions for all the parts.

0018-9162/89/0300-0021$01.00©1989 IEEE


Step 1


Step 2

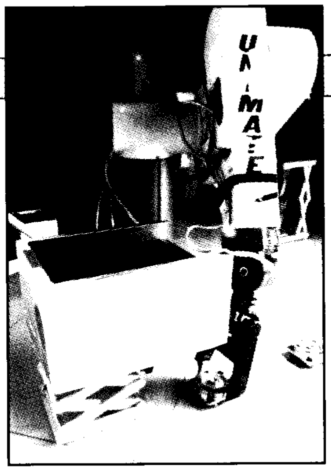
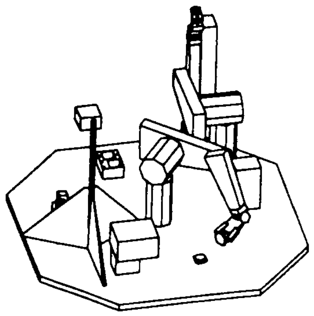
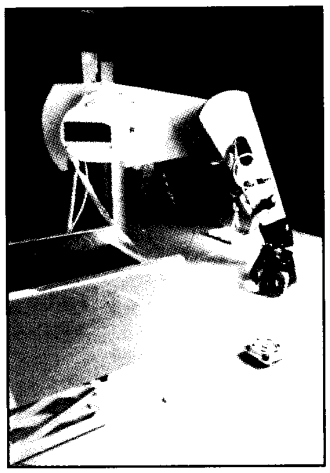
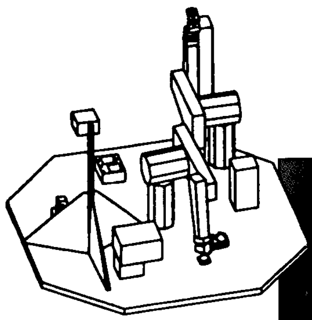
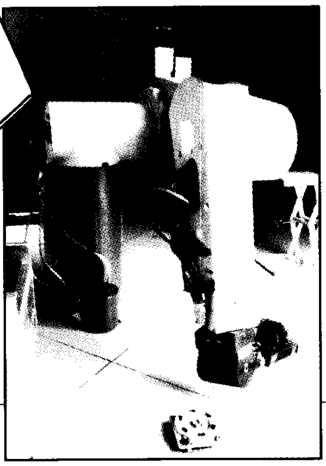


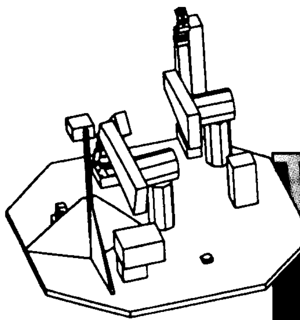
Figure 1. Steps 1-5 in a pick-and-place operation planned by Handey. Photographs courtesy of Francisco Garcini.

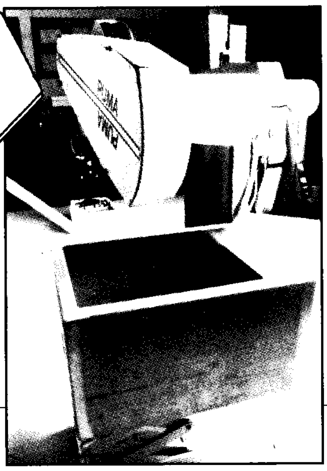

Step 3




Step 4




Step 5



Then, for each assembly step:

- Choose a grasp on the part.
- Plan a motion to grasp the part.
- Plan a motion to the assembly location for the part.
- Plan a motion to extract the gripper.

The pick-and-place problem corresponds to the last four steps above.

Ideally, the assembly problem will reduce to a number of independent subproblems, roughly one per line in the characterization above. Unfortunately, this is only true in extremely simple cases. In fact, the solutions to all these problems are tightly linked:

(1) The optimal initial placement of the parts depends on the order of assembly.

(2) The order of assembly depends on the feasibility of certain assembly steps, for example, whether a subassembly can be inserted into the main assembly in one piece or whether it needs to be assembled in position.

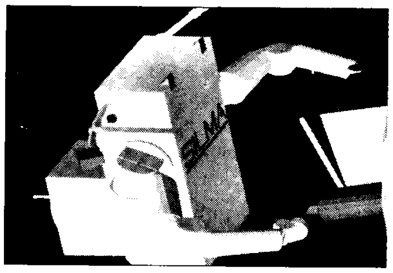
(3) The choice of a grasp on a part depends on the environment at the initial location, which depends on the initial placement of nearby parts. The grasp also depends on the placement of parts at the target, which depends on what previous assembly steps have been performed. The choice of grasp depends on the range of motions of the robot; surprisingly many potential grasps simply will not be reachable.

(4) The choice of a grasp (and ungrasp) motion depends on the choice of a grasp and on the whole environment, especially near the grasp and destination.

(5) The choice of an assembly motion depends on whether a grasp compatible with the assembly has been found; otherwise, the part must be regrasped.

In short, all the subproblems of the assembly problem are interdependent. This also holds true for the pick-and-place problem we are considering.

One possible approach to the pick-and-place problem, given the interdependence of the decisions, is to treat it as a single motion-planning problem with special



constraints. In particular, the effective shape of the robot changes once it grasps the object, so the constraints on its motion change. The problem with this approach is that it runs into significant computational complexity; the additional degrees of freedom in the choice of a grasp add to the robot's motion freedoms. But, we expect all complete algorithms for motion planning to have worst-case time-complexity that is exponential in the degrees of free-

Configuration space

Robot manipulators are articulated devices made up of a series of rigid *links* connected by one-degree-of-freedom *joints*. Joint motions are either rotational or translational. The positions of all parts of a rigid robot are completely specified by the values of the joint parameters, known collectively as the *joint angles*. Many robots support an alternative specification for desired position, namely, the position and orientation of the robot gripper. But, gripper position is not a unique specification of the complete robot position; many sets of joint angles can exist that place the gripper in the same location.

Any set of parameters that uniquely specify the position of every part of a system is called a *configuration*, and the space defined by those parameters is the *configuration space* or *C-space*. Most algorithmic approaches to motion planning require a characterization of those configurations of the robot that cause collisions (or the complement of that set). We call the collection of configurations that produce collisions the *C-space obstacles*.[8]

Consider the following simple case (see Figure 2a). In this example, we

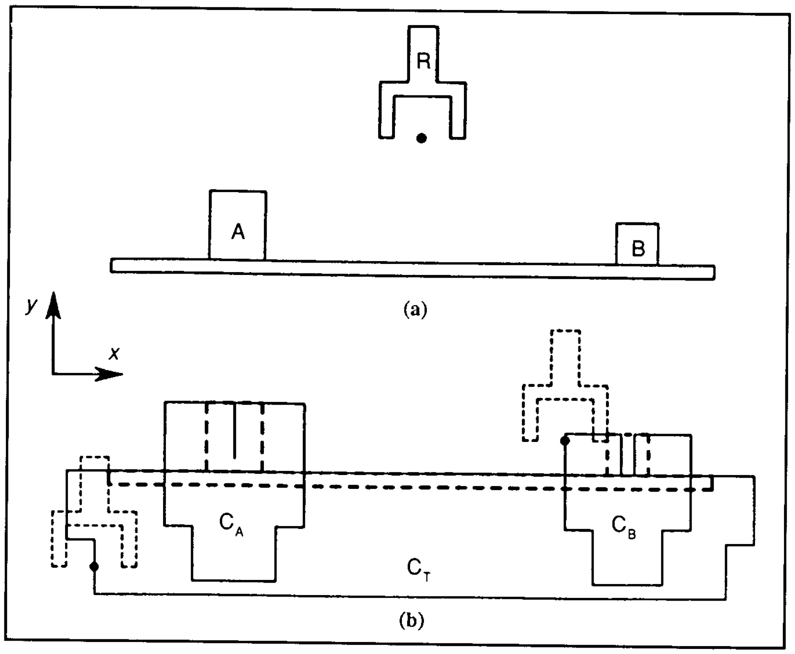


Figure 2. Cartesian and C-space obstacles. (a) A scene with three obstacles, *A*, *B*, and the table. The disembodied robot hand *R* is capable only of displacements along the *x* and *y* axes. (b) The C-space for this hand is the *x,y* plane. The C-space obstacles corresponding to *A* (C_A), *B* (C_B), and the table (C_T) are seen. The gripper is shown (in a fine dashed line) at two points of the C-space obstacle boundary.

dom but polynomial in the size of the environment (Reif has shown motion planning to be PSPACE-hard[5]). Therefore, any practical solution to the pick-and-place and assembly problems must involve decoupling and other forms of dimensionality reduction (see the accompanying sidebar, "Configuration space").

Approximate approaches to the pick-and-place problem

The crucial step in solving the pick-and-place problem is choosing how to grasp the part. Once this choice has been made, the problem boils down to separate motion-planning problems in the robot's C-space. But, we cannot choose a grasp simply by looking at the environment near the part; we must consider the effect of the grasp on the possibility of finding a path to the goal. Therefore, any attempt at decoupling must consider constraints imposed on the choice of grasp by the environment at the goal as well as that at the origin.

One necessary condition that the choice of grasp point must satisfy is that there be no collisions of the robot with any object at either the initial or final position of the part. In addition, there must be some path connecting these initial and final positions along which there are no collisions. In practical situations, however, given safe initial and final destinations, a path can usually be found.

Given this heuristic assumption about the availability of paths, there only remains the problem of finding a grasp that is reachable for the part in its initial position and avoids collisions for the initial and final positions. There are at least two ways of doing this:

(1) Characterize the reachable grasps at the initial grasp position, then characterize the grasps that cause no collisions at the destination. The grasps in the intersection of these two grasp sets are collision-free at both positions and reachable at the initial position.

(2) Compute the transformation T that maps the grasped part from its initial to its final location. Apply the inverse transformation T^{-1} to a copy of the obstacles near the final position of the part. Add these transformed obstacles to the obstacles near the initial position of the part. Find a path to any legal grasp that avoids both sets of obstacles (see Figure 4).

These two methods are not equivalent; the second method will, in general, find fewer grasps than the first method will. In finding a path that avoids both sets of obstacles, we have constrained the problem beyond what is strictly necessary to guarantee finding a grasp that is reachable at the initial position and safe at the final one. Recall that we are planning the initial grasp approach and not the path to the final destination. It is not necessary, therefore, that the complete approach path avoid the obstacles derived from the destination, only that the *chosen grasp*, that is, the final position in the approach path, avoid the obstacles.

We chose the second method for Handey

are limited to a disembodied robot hand capable only of displacements along the *x* and *y* axes; we assume that the finger opening remains fixed to simplify the C-space. The C-space for this hand is the *x,y* plane. The C-space obstacles corresponding to objects *A* and *B* and the table in Figure 2a are the objects bounded by dark lines in Figure 2b. Every *x,y* point inside one of the C-space obstacles represents an *x,y* position of *R*'s reference point (the dark circle in Figure 2a) that causes a collision. C_A are the configurations that cause a collision between *R* and *A*; C_B are the configurations that cause a collision between *R* and *B*; C_T are the configurations that cause a collision between *R* and the table.

Consider a simple robot with two rotational joints, whose joint angles are θ_1 and θ_2 (see Figure 3a). A point in the C-space specifies both joint angles and, therefore, the position of every part of the robot. The set of configurations for which the robot is in collision with an obstacle defines C-space obstacles. In this sense, the C-space for the robot is analogous to that of the disembodied hand above, but the C-space obstacles of the robot are significantly more complex than the ones for the hand,[8] (see Figure 3b). Note that the C-space parameters of the robot are angles. Therefore, the line at the top of the diagram, $\theta_2 = 2\pi$, is the same line as the bottom of the diagram, $\theta_2 = 0$; we can say the same for the left and right lines, $\theta_1 = 2\pi = 0$. We can wrap the diagram into a tube so that the top and bottom lines meet, then we can wrap that tube into a doughnut (torus) to make the left and right ends meet. Thus, the C-space for the two-link robot is really the surface of a torus.

Finding a collision-free path between two specified configurations requires finding some path in the C-space that connects the two specified configurations and does not penetrate any of the C-space obstacles. Doing this requires characterizing the space outside of all of the C-space obstacles, possibly by a decomposition into disjoint cells. A sample path between the two arm configurations shown in Figure 3a is shown in the C-space of Figure 3b.

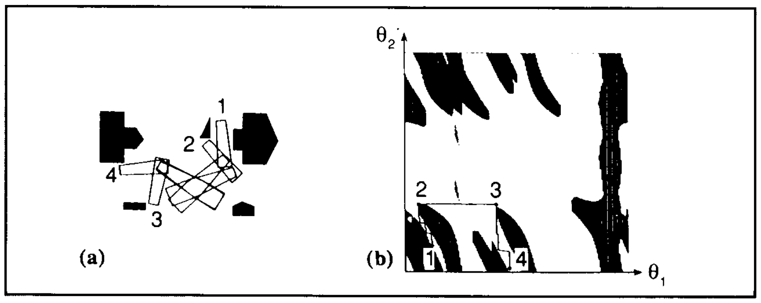


Figure 3. C-space obstacles for a simple robot. (a) A robot with two joint angles, θ_1 and θ_2, and its obstacles. (b) The corresponding C-space obstacles (the hatched regions). The joint angles are also the labels on the axes of the C-space. A safe path in this C-space is shown, together with four configurations of the arm along the path.

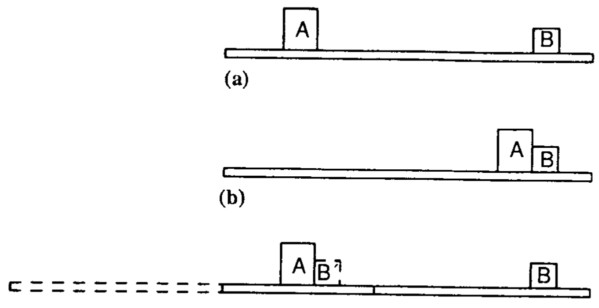
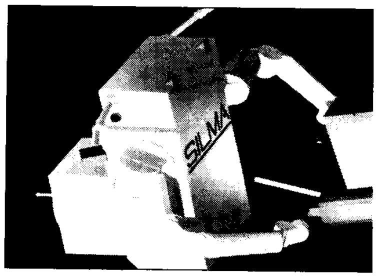


Figure 4. A simple pick-and-place problem: (a) the initial position of A; (b) the final position of A; and (c) the pick-and-place example showing the obstacles for the final position of part A (dashed lines) superimposed on the obstacles for the initial position (solid lines).



because it does not require characterizing sets of grasps, whether simply collision-free or reachable. Computing these sets can be a burdensome task. The second method requires only the ability to find a path from a known initial position to some point within a constrained set (legal grasps).

In many cases, no grasp exists with an approach path that avoids obstacles from both the origin and the destination. In this case, Handey plans a sequence of regrasping steps (placing the object on the table and then regrasping it) that results in a grasp compatible with the target position.[6] Once again, we use only the necessary condition that the grasp does not cause a collision at the origin or destination.

Here is a more complete outline of the solution of the pick-and-place problem in Handey:

(1) Enumerate the possible distinct types of grasps. Handey operates in the domain of polyhedral models and robots with parallel jaw grippers; therefore, the distinct grasps are limited to pairs of part features (either faces, edges, or vertices) that can be in simultaneous contact with the parallel interior faces of the gripper. Our implementation is currently limited to pairs of parallel faces.

(2) Rank the potential grasp faces by the percentage of the area of intersection of the faces that remains unobstructed after projecting onto the face any obstacles near the face, at both origin and destination.

(3) Select a potential pair of grasp faces, and pick target points for the fingertips in the unobstructed sections of the faces.

(4) Plan a path for the robot gripper, constrained to move parallel to the grasp faces, from a nearby safe point (chosen arbitrarily) to a point close to the chosen grasp point for which there is sufficient overlap between the gripper and the grasp faces. If no path can be found, select a different grasp face and try again. The path must avoid collisions and keep the robot within its legal range of motion.

(5) Plan a collision-free motion (using a low-resolution approximation of C-space) for the robot from its current position to the arbitrarily chosen point near the initial grasp.

(6) Plan a collision-free motion for the robot, carrying the grasped part (using a low-resolution approximation of C-space) from the initial grasp position to the specified target position.

(7) When it is not possible to find a grasp that avoids collisions at the initial and target positions, plan a sequence of regrasping steps that produce a grasp compatible with the target position. For each required grasp and placement, plan collision-free motions for the robot.

The crucial steps in this formulation rely on finding collision-free paths between two known points or between a known point and a target set.

The worst-case complexity of motion planning is exponential in the number of degrees of freedom of the robot; the best motion-planning algorithm for a d-degree-of-freedom robot has time complexity $O(n^d \log n)$, where n is the product of the number of faces, edges, and vertices in the obstacles.[7] Although this bound is polynomial in the environment size, n, this complexity still makes complete motion-planning algorithms for real robots impractical on typical serial computers. We also found that pick-and-place planning has a higher dimensionality than simple motion planning. Therefore, we have pursued a heuristic decoupling strategy for pick-and-place planning. To obtain good running times, we must also pursue heuristic strategies into the motion-planning subproblems.

Heuristic motion planning in Handey

Earlier, we outlined an approach to motion planning based on computing C-space obstacles; Lozano-Pérez describes this approach in more detail.[8] Although we could use the approximate algorithms described there to solve directly the six-degree-of-freedom motion-planning problems required for the typical pick-and-place problem, the running times would be too large for practical use, even in experimentation. Instead, we have adopted several heuristic methods of reducing both the dimensionality and the average size of these motion-planning problems:

(1) Using a local motion planner for small motions near obstacles together with a low-resolution C-space for large motions farther away from obstacles. The local planner simulates the effect of a body being acted on by repulsive forces from the obstacles and attractive forces from the goal. This general approach is known as the *artificial-potential-field approach.*[9]

(2) Limiting the large motions to the

first three joints of the robot, but with the ability to change the values of the last three joints.

(3) Approximating arbitrary polyhedral obstacles by obstacles with a constant cross section that simplify the computation of the low-resolution C-space obstacles.

The combination of these strategies has significantly reduced the running times for motion planning (from a few minutes to perhaps 30 seconds) without significantly reducing Handey's ability to solve pick-and-place problems. We describe these methods in the following subsections.

Local planning: the quasipotential method. In Handey, motion near an obstacle happens primarily when the object is being grasped. In those circumstances, the robot's motion is constrained so that the gripper moves in the plane of the chosen grasp faces. This means that we have a three-degree-of-freedom planning problem instead of a six-degree-of-freedom problem. But, those three motion freedoms correspond to Cartesian motions of the gripper in the grasp plane (x,y,θ) and not to individual joint motions of the robot arm. Exploiting this reduction in dimensionality requires a different planner from the one we use for the gross motions of the arm.

We could construct approximate x,y,θ C-space obstacles for the gripper moving in the grasp plane and plan the gripper motions in this space. But, instead, we have adopted a local, incremental planner that does not build an explicit C-space. We chose this strategy for two reasons:

(1) In Handey, some of the obstacles might not be modeled directly as polyhedra by the system; instead, they might be represented implicitly by an array of depth measurements. We could have tried to build a polyhedral approximation to this data and use the existing C-space obstacle algorithms for polyhedra. Instead, we chose a method that deals with the data more directly.

(2) Typical cases of approaching obstacles within the grasp plane are simple, and a nearly direct path exists. Incremental methods have a lower computational overhead in these situations. In difficult cases, local, incremental methods fail to find an answer. In those cases, we can fall back on guaranteed methods.

Traditional potential methods[9] measure the distance between a number of points on the moving body (the gripper in our case) and obstructions in the grasp plane, compute repulsive forces proportional to a power of the distance, and sum them with an attractive force based on the distance from the gripper to the goal. These total artificial forces and torques acting on the gripper are then used to compute a motion for the gripper via the simulated dynamics of a viscous damper, $f = bv$, where f is the total force/torque vector and v is the resulting velocity vector.

Our version of this method effectively uses a potential that is a high power of the distance. Beyond some gripper-to-obstacle distance d, the force on the manipulator is 0; within that distance, the force is such as to prohibit motion toward the obstacle. A grid represents the motion plane, and obstacles correspond to filled cells in the grid. The starting point is some point on the edge of the grid in the same connected component of the grid as the target point.

Surrounding the gripper at a distance d and moving with it are *bump lines*, that is, line segments on the grasp plane that are checked each iteration for collisions with filled grid cells (see Figure 5). A *bump vector* is a unit vector perpendicular to a bump line pointing away from the gripper.

In the absence of intervening filled grid cells in the motion plane, the gripper's motion will be a simple translation along the vector connecting the finger and target point. We call the unit vector in this direction the *goal vector*.

After investigating all the bump lines for collisions with filled cells, we construct a unit circle and map onto it the goal vector and the bump vectors whose correspond-

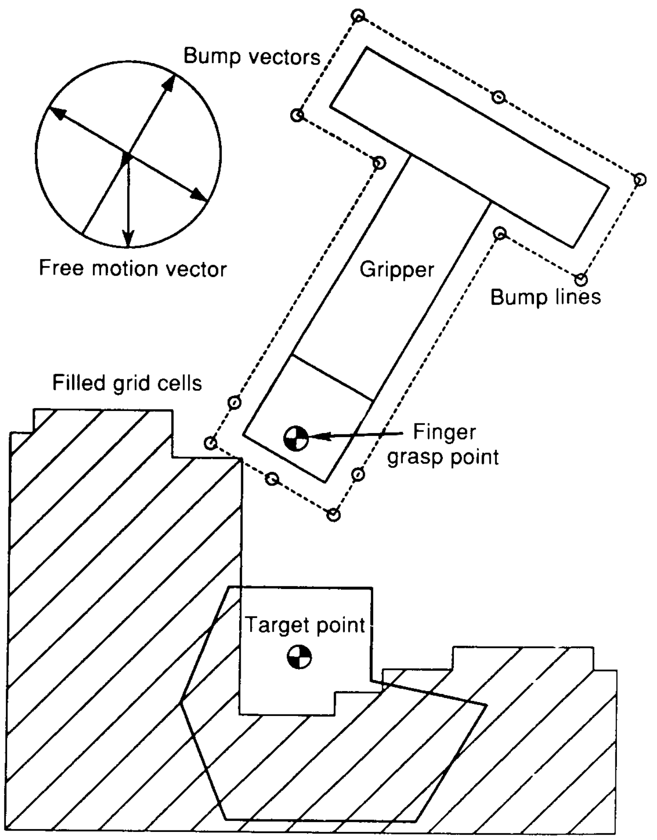


Figure 5. Illustration of the definitions in the quasipotential method.

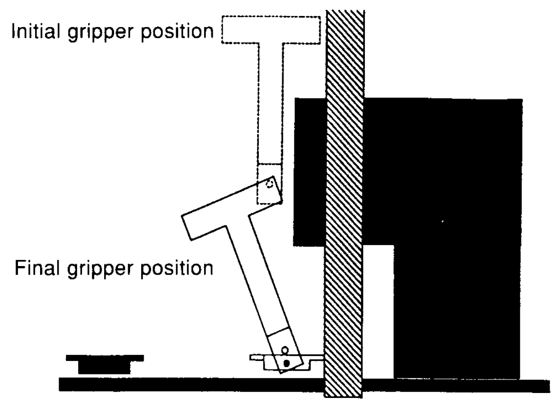
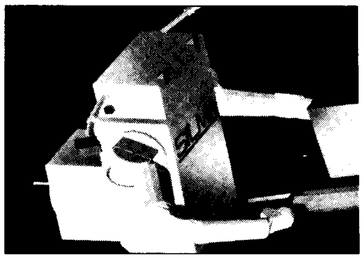


Figure 6. The initial and final position of the gripper for the initial grasp of the example in Figure 1. The dark obstacles are present at the pickup point, the hatched obstacle is a copy of the table at the destination where the regrasp is to be done.



ing bump lines have not detected collisions. Figure 5 shows a typical situation.

To compute the translational motion of the gripper, we compare the goal vector with the multiplied bump vectors. If the goal vector has no component in the direction of a zero-length bump vector, the gripper moves in that direction. Otherwise, as in the figure, the gripper moves along the nonzero bump vector closest in direction to the goal vector.

The bump lines also provide a convenient way to compute a torque to rotate the gripper. Any colliding bump line produces a torque whose magnitude is proportional to the cross product of the bump vector and a vector connecting the *finger grasp point* with the center of the bump line. The total torque on the gripper is just the sum of the torques generated by each colliding bump line.

The gripper is free to rotate about the finger grasp point (shown near the finger tip in Figure 5). Rotations and translations are limited in such a way that no point on the gripper moves by more than one grid cell per iteration. This precludes the possibility that any filled grid cells will penetrate the bump lines during an incremental motion.

At each iteration, a check ensures that a motion in the computed direction will yield a reachable gripper position. If the position is not feasible, then a different direction (if any remain) is tried.

Every few iterations, the position of the gripper is compared to a previous position. If no significant progress has been made, the path is terminated. At this point, the position of the gripper in relation to the target area is checked. If the fingers sufficiently overlap the target (for example, the chosen faces of a grasp), then the path is accepted and returned; otherwise, the planner either tries to plan a path from another starting point or tries another pair of faces.

Figure 6 shows initial and final positions obtained by the quasipotential method for the initial grasp of the example in Figure 1. Note that the finger grasp point cannot actually reach the chosen target point, but that the grasp sufficiently overlaps the target face.

Although this method does not guarantee a solution, it has performed quite well in most of the cases we have encountered. We believe we can enhance the method's performance by improving the choice of starting point and avoiding a reliance on a fixed attraction point. We are currently experimenting with a variation of the method that picks a starting point by finding the maximal clearance translational paths from the target point to the edges of the motion grid. The starting points are the ends of the paths. During each iteration, the attraction point used for computing the artificial forces acting on the gripper will move along the chosen path. We hope this will avoid many of the problems of local minima and nonconvex obstacles.

Global planning: low-resolution C-space. The local quasipotential method is effective near the target when we know that a nearly direct path exists. Local methods are less effective for gross motions that span a significant segment of the work space, so we use a low-resolution C-space method for these motions. Although the basic method is that described in Lozano-Pérez,[8] the implemented method uses several heuristics to lower the average computation time.

One key approximation limits the C-space obstacle construction to the first three joint angles of the robot. The planner builds three three-dimensional *slices* of the underlying six-dimensional C-space. One slice is built with the wrist angles fixed at their value at the start of the path, another slice with the angles set at their value at the end of the path, and the last slice for the range of wrist angles between the start and the end. The free-space representation in these three slices is linked into a single free-space representation that can be searched for a path.

We build the first two slices at high resolution (quantization of one degree), but make no attempt to build the full three-dimensional slice of the C-space. In fact, we make every attempt to limit the size of the slices. The idea is to move as close as possible to the initial and goal points within the slice that spans the orientations of the wrist between the initial and goal points. Then, only the final segments need to be traveled in the remaining two slices. The computation of these slices, therefore, proceeds on an "on-demand" basis. First, we build the slice for a very narrow range

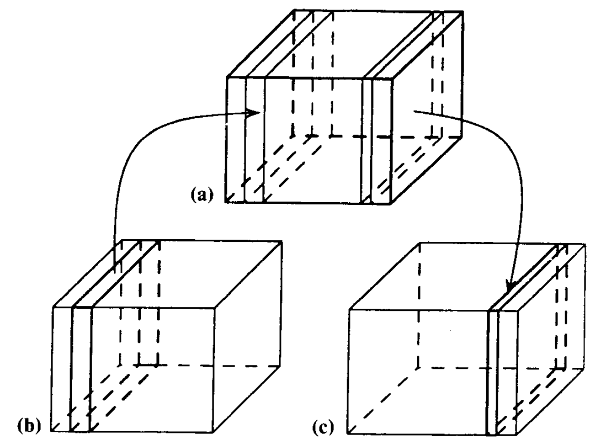


Figure 7. The gross-motion-planning problem is split into three three-degree-of-freedom slices: (a) a low-resolution C-space slice for the total range of values of the wrist angles, (b) a high-resolution C-space slice for the initial values of the wrist angles, and (c) a high-resolution C-space slice for the final values of the wrist angles.

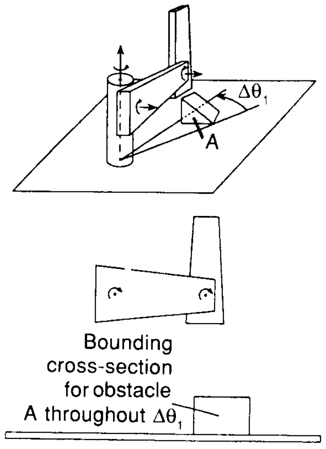


Figure 8. Obstacles can be approximated by tori, centered at the base of the robot, with constant polygonal cross sections in the plane of the second and third links of the robot.

of the first three angles. We stop if we find a path in this range; otherwise, we widen the range incrementally until we find a path. This strategy is illustrated in Figure 7.

The third slice, spanning the range of wrist angles for the motion, is built to span the whole range of the first three joint angles. But, we build this slice at lower resolution (eight degrees) and approximate the robot and the obstacles to simplify the computation. This simplification exploits the fact that the computation of C-space obstacles for planar robots and obstacles is much more efficient than that for solid polyhedra.[8] We can take advantage of the fact that the Unimation Puma 560 robot used in Handey has, as do many commercial robots, two links with parallel rotation axes (see Figure 8). Therefore, we can approximate the arm as a planar two-link arm operating in a plane determined by the first joint angle. Then, we can approximate the obstacles by tori, centered at the base of the robot, with constant polygonal cross sections in the plane of the second and third links of the robot (see Figure 8).

The result of this process approximates the three-degree-of-freedom C-space using only planar computations. Although this does not affect the asymptotic complexity of the method, it results in a significant speed up.

The interaction between local and global planning. Given a problem that requires moving from a position close to an obstacle to another position close to an obstacle, such as a regrasping step, Handey proceeds as follows:

(1) Use the quasipotential planner to move away from the initial position very near an obstacle. Call this point I'.

(2) Use the quasipotential planner to move away from the final position very near an obstacle. Call this point F'.

(3) Use the C-space planner to move from I' to F'.

The current implementation of Handey does not choose the points I' and F' with the gross motion planner in mind. In particular, there is no guarantee that these points will be in the low-resolution free-C-space built by the gross motion planner. This is not a requirement, but it minimizes the computation required to plan the complete path. We could make this connection by marking as desirable targets those grid points (specifying the gripper position and orientation) that map into free-C-space points (specifying the first three joint angles of the arm). Nevertheless, this hybrid local/global approach to planning motions has proven quite effective in our experiments.

Handey is unique for being the first task-level system that has been extensively tested in relatively complex tasks. Since Handey operates by locating an obstructed object placed in a random orientation and taking it to any specified location, it must handle quite general motion-planning problems. But, to be thoroughly tested, it must run in reasonable time. This conflict has forced us to search for simplifying assumptions that do not significantly reduce the system's generality. We believe that this search will be critical to the development of task-level robot systems. Handey's capabilities are only a small fraction of those required for a comprehensive system. Other important capabilities are

- planning force-controlled motions to assemble objects in the presence of position error;
- planning coordinated motions for multiple robots;
- planning the placement of the parts in the work space to optimize task execution; and
- planning the nature and order of operations required to carry out a task, for example, the order in which to assemble parts.

We are currently investigating all these capabilities within the framework of Handey. □

Acknowledgments

This work was funded primarily by the Office of Naval Research under contracts N00014-85-K-0214 and N00014-86-K-0685. Additional support was provided by a National Science Foundation Presidential Young Investigator Award (Lozano-Pérez), the French Centre National de la Recherche Scientifique (Mazer), and Digital Equipment Corporation. In addition to the authors, the following people have contributed to the development of Handey: Pierre Tournassoud of INRIA (France), Alain Lanusse of ETCA (France), and Eric Grimson of MIT. The visits of Lanusse and Tournassoud to MIT were funded by ETCA and INRIA, respectively.

References

1. H.A. Ernst, *A Computer-Controlled Mechanical Hand*, PhD thesis, MIT, Cambridge, Mass., 1961.

2. T. Lozano-Pérez and R.H. Taylor, "Geometric Issues in Planning Robot Tasks," to be published in *Problems of Robotics*, MIT Press, Cambridge, Mass., 1989.

3. M.T. Mason, "Automatic Planning of Fine Motions: Correctness and Completeness," *Proc. IEEE Int'l Conf. Robotics and Automation*, 1984, CS Press, Los Alamitos, Calif., Order No. 526, pp. 492-503.

4. D.E. Whitney, "Quasistatic Assembly of Compliantly Supported Rigid Parts," *ASME J. Dynamic Systems, Measurement, and Control*, Vol. 104, March 1982, pp. 65-77.

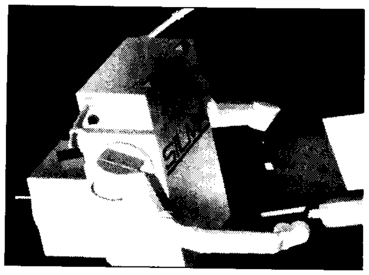
5. J.H. Reif, "Complexity of the Generalized Mover's Problem," in *Planning, Geometry, and Complexity of Robot Motion*, Ablex Publishing, 1987, pp. 267-281.

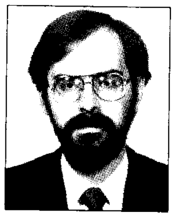
6. P. Tournassoud, T. Lozano-Pérez, and E. Mazer, "Regrasping," *Proc. IEEE Int'l Conf. Robotics and Automation*, 1987, CS Press, Los Alamitos, Calif., Order No. 787, pp. 1,924-1,928.

7. J.F. Canny, *The Complexity of Robot-Motion Planning*, MIT Press, Cambridge, Mass., 1987.

8. T. Lozano-Pérez, "A Simple Motion-Planning Algorithm for General Robot Manipulators," *IEEE J. Robotics and Automation*, Vol. 3, No. 3, June 1987, pp. 224-238.

9. O. Kathib, "Real-Time Obstacle Avoidance for Robot Manipulator and Mobile Robots," *Int'l J. Robotics Research*, Vol. 5, No. 1, Spring 1986, pp. 90-98.





Tomás Lozano-Pérez is an associate professor of computer science and engineering at the Massachusetts Institute of Technology, where he is a member of the Artificial Intelligence Laboratory. His research interests are in robotics and artificial intelligence; he teaches courses in these areas at MIT. Before joining the MIT faculty in 1981 he was on the research staff at IBM's T.J. Watson Research Center during 1977 and the MIT AI Laboratory during 1974 and again during 1980. He has been co-editor of the *International Journal of Robotics Research*; co-editor of the book *Robot Motion* (MIT Press, 1982); program chairman of the 1985 IEEE International Conference on Robotics and Automation; and recipient of a 1985 National Science Foundation Presidential Young Investigator Award. He is member of the ACM and the IEEE Computer Society.

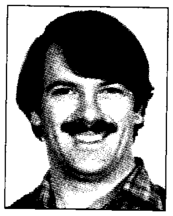
Lozano-Pérez received his BS in 1973, his MS in 1976, and his PhD in 1980, all from MIT in computer science.

Emmanuel Mazer is a research fellow of the Centre National de la Recherche Scientifique (CNRS) working at the Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle (LIFIA) in Grenoble, France. His current interests are highly redundant manipulators, nanomanipulators, and automatic robot programming. Mazer has been visiting scientist at the Artificial Intelligence Laboratory of the University of Edinburgh during 1981-1982, research fellow at IMAG (Grenoble) during 1982-1984, technical director of ITMI Marketing during 1984-1986, and visiting scientist at MIT during 1986-1988.

Mazer received a master's degree in applied mathematics in 1979 and a PhD in computer science in 1988, both from Grenoble (INPG).

Joseph L. Jones has been a research engineer at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology for six years. His interests include task-level planning and mobile robots. Previously, Jones worked as an engineering physicist at the Bates Linear Accelerator.

Jones received BS and MS degrees from MIT in 1975 and 1978, respectively.



Patrick A. O'Donnell is currently a research engineer at the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. His current interests include robotics planning, massively parallel computing, and simulation of biological structures. He is a member of ACM, the IEEE Computer Society, the IEEE Robotics and Automation Society, Tau Beta Pi, and Eta Kappa Nu.

O'Donnell received the BS and MS degrees from MIT in 1983.

Readers may contact Lozano-Pérez at the MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, Mass. 02139.