Solving Jigsaw Puzzles by a Robot

GRIGORE C. BURDEA, MEMBER, IEEE, AND HAIM J. WOLFSON, MEMBER, IEEE

Abstract—**An integrated vision-manipulation algorithm for assembly of *apictorial* jigsaw puzzles is presented. The paper discusses the solution of large jigsaw puzzles using vision, combinatorial optimization, and fine assembly techniques. The implementations of a vision algorithm for assembly of large jigsaw puzzles, and a fully integrated robotic-vision algorithm for assembly of small puzzles are presented. The problem of assembling pieces with unpredictable shape is discussed.**

I. INTRODUCTION

ASSEMBLY and repair are major robotic application tasks. Much work has been done to investigate assembly of parts with relatively simple shapes (see [1]–[5]). The famous "peg in the hole" problem [6]–[11] was intensively investigated for such shapes. Less work was done on assembly of parts with complicated and *a priori* unpredictable shapes. The decision to tackle the jigsaw puzzle assembly problem was made, since we regarded it as a strenuous test to our capabilities both in machine vision and robotic assembly. Here, not only the shapes of the different pieces are complex and *a priori* unknown, but also the correct mating of the pieces is unknown, and has to be solved using the geometric shape information. This problem is even complicated for humans, since we solved an *apictorial* [12] or "white" puzzle, where no picture is given to facilitate the solution. Only the shape of the different pieces can be considered. As far as we know, the first algorithm to solve *apictorial* jigsaw puzzles was proposed by Freeman and Garder [12], who successfully solved a 9-piece jigsaw puzzle using visual information only. Although one may argue that in most industrial assembly tasks the problem is usually less complicated, and much of the information can be given in advance, there are obvious applications of the *puzzle assembly*. For example, repair of broken objects or restoration of archeological findings is a *puzzle assembly* problem.

There are numerous problems which have to be tackled in *puzzle assembly*. First, and for humans the most complicated, is the puzzle solution itself. An approach which is usually taken in this case by a human operator is to conjecture a mating between two pieces based on the visual information of their shape, and then try to assemble these two pieces along their supposedly matching boundary. In case the actual assembly fails another match is conjectured and checked, and so far. In this straightforward approach we already see a close interaction between the visual tasks and the manipulation tasks. We use vision to identify the different pieces and to conjecture their mutual match. Next we use our hands to grasp a piece, move it to a position where it fits the other (the position was "computed" by vision), and then make small adjustments of fine assembly. Finally, using both vision and touch we decide whether the parts actually fit. We have tackled the above mentioned problems, and the proposed solutions are described both in Section II-B, which deals with the visual information extraction and the "local" visual best possible matching of two puzzle pieces, and in Section III, which describes a fully implemented robotic assembly (see also [13]) of a small number of puzzle pieces integrating both the visual and tactile aspects of the solution.

The machine vision module is used to detect puzzle pieces in the work area, and for each pair of such pieces, to find a rigid motion giving the best possible least squares (see Section III-B) match of the boundaries of these pieces. If the least squares score of such a match is below a certain threshold, the actual assembly is attempted. The piece interlocking is done using force feedback based fine matings. In the *assembly* stage we are faced with the problem of fine assembly of parts with quite complicated shapes. Although these shapes belong to a certain class of typical *jigsaw puzzle pieces* (see Fig. 2), their exact shape is *a priori* unknown. This unpredictability of shapes excludes the use of special fixtures to facilitate the assembly task. The assembly module has also to make a decision, whether the proposed matching is feasible, thus being able to distinguish between local errors in the fine assembly that can be corrected and between total incompatibility between the pieces. Since our puzzle pieces are actually three-dimensional (3-D) (although of constant height) these local errors involve possible 3-D translations and rotations. In that sense we have to tackle a more complex case than previous work that analyzes only 3-D translational errors (see [3]).

The above discussed implementation mimics the human puzzle solution and assembly technique. However, this "human-like" approach might be quite inefficient, when one is faced with an assembly of a large *apictorial* puzzle with pieces having visually quite similar shapes (see Figs. 2–5). In such a case, a mating conjecture based on purely local shape comparison might be inaccurate, since the distortion of the visual data acquisition step may result in a wrong (although

Manuscript received June 12, 1988; revised January 14, 1989. This work was performed while the authors were at the Robotics Research Laboratory, Courant Institute of Mathematics, NYU, and was supported by the Office of Naval Research under Grants N00014-82-K-0381 and N00014-85-K-007 Work Unit NR 4007006, and the National Science Foundation under Grant NSF-DCR-83-20085.

G. C. Burdea is with the Department of Electrical and Computer Engineering, College of Engineering, The State University of New Jersey—Rutgers, Piscataway, NJ 08855-0909.

H. J. Wolfson is with the Robotics Research Laboratory, Courant Institute of Mathematical Sciences, New York University, New York, NY 10003. He is also with the Computer Science Department, School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel.

IEEE Log Number 8930464.

1042-296X/89/1200-0752$01.00

quite similar) piece achieving the best "matching score." Hence, a significant number of the matings suggested by the vision module will be rejected by the assembly module, resulting in a very time-consuming verification. This is, usually, the case in a human solution of a large *apictorial* puzzle, when a significant number of puzzle piece matings is decided by a very rough visual comparison and a subsequent verification by direct assembly. Even in the latter case, where we use our superb and fast fine assembly capabilities, enhanced by vision, the solution takes an impractical amount of time. To overcome this difficulty, we turn to a *global* solution of the puzzle assembly problem (see Section II-C). The rationale for this approach is based on the fact, that although *locally* some incorrect solutions may score higher than the correct one, the *global* score of the correct configuration will probably be much higher than of a wrong one. This *global* approach has been first applied to the solution of large jigsaw puzzles using vision only (see [14]). There we successfully solved 100-piece puzzles (see Figs. 3 and 5).

Earlier approaches to the solution of jigsaw puzzles by computer vision [12], [15] were based only on local piece mating considerations. They successfully tackled assembly of small puzzles (about 10 pieces in [12], 4 pieces in [15]). The above mentioned algorithms did not assume any knowledge of the frame pieces of the puzzle, which is exploited in our approach. On the other hand, they did require nonmatching puzzle sides to be significantly different from each other. Also, in an earlier experiment the *local* matching algorithm of Section II-B was applied with some backtracking to a solution of a 16-piece puzzle with quite similar shapes. The performance of such *local* techniques is usually limited to small puzzles and can be significantly improved by a *global* approach.

In this paper we emphasize the fact that even the latter *global* solution, which is based on combinatorial optimization techniques, can strongly benefit from a robotic interaction. One can not only verify a proposed solution by actual robotic assembly, but a truly integrated process can be developed (see also [16]). In Section II-C we discuss our integrative approach (see Fig. 1) combining application of machine vision techniques for local curve matching estimation, mathematically sophisticated combinatorial optimization techniques to get global solution proposals, and robotic fine assembly techniques, which are used to verify these solutions and feed back partial results, thus enabling significant speedup in the iterative application of the previous modules. The difference between this *global* approach and the one discussed before is that we use the robotic verification step only to verify suggested global solutions, and not the local ones. This will bring a significant reduction in the application of the time-consuming verification step.

This paper is organized as follows. Section II describes the machine vision algorithm for solution of large jigsaw puzzles, its experimental results, and the proposed integrative method using robotic assembly feedback. Section III describes the integrative robotic assembly system for solution of small jigsaw puzzles and its practical implementation by an IBM 7565 (RS-2) Cartesian robot. It focuses on the solution of the grasping and manipulation problems involved in fine assembly, and in coping with uncertainty. Section IV states some future improvements and research directions.

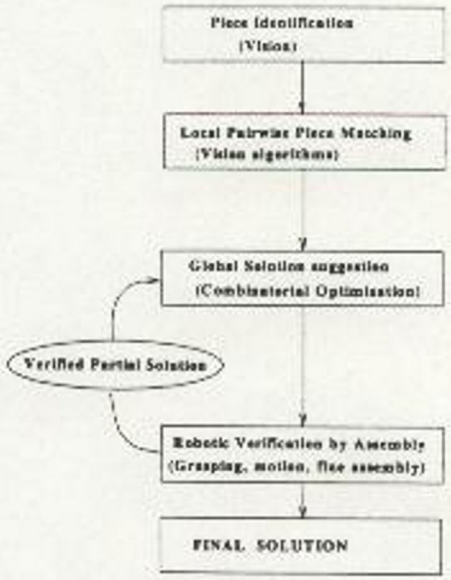


Fig. 1. The integrated global solution scheme.

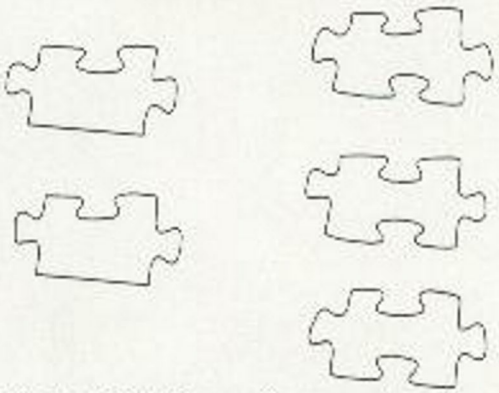


Fig. 2. Different frame and interior puzzle pieces.

II. The Vision Algorithm

In this section we preent the vision algorithm for assembly of large jigsaw puzzles. The proposed integrative solution including robotic assembly feedback is described in Section II-C. The *global* solution algorithm using vision only has been successfully implemented (see Figs. 3 and 5).

Usually puzzle assembly is considered as a strenuous test for curve-matching algorithms [15]. We have used such an algorithm due to Schwartz and Sharir [17], which proved to be very robust. However, in our case, the solution has been complicated by two major factors. The first one is the large number of puzzle pieces to be assembled. Previous work [12], [15] dealt with small puzzles. The second one is the shape similarity of different puzzle pieces (see Fig. 2). This made correct decisions, based on *local* best match of two pieces, almost impossible. Hence we introduced combinatorial optimization techniques to achieve a *global* solution. The section describes this technique. The reader is referred to [14] for additional details on the implementation of the global puzzle solution using vision only, and a solution to the problem of

simultaneous assembly of several puzzles. We also emphasize the benefits of integration of robotic assembly and verification stages in the *global* solution scheme.

It should be noticed that finding a nonheuristic efficient puzzle assembly algorithm might be impossible. Specifically, the related *square-tiling* problem is known to be *NP-complete* [18]. There, one is given N^2 rectangular tiles with colored sides (the number of different colors is not less than N), and the task is to find whether there is a tiling of an $N \times N$ square such that only sides of equal colors are matched. There is a close relation between this problem and the puzzle assembly problem since a specific side of a puzzle piece can be considered as a *geometric color*. Consequently, the algorithms proposed in this section, which apply fast heuristic techniques, will not always lead to the discovery of the correct solution. The integration of these techniques with a robotic assembly verification should ensure convergence to the correct solution.

A. Definition of the Problem

We consider the solution of an *apictorial* large jigsaw puzzle. Specifically, we will refer to the examples of Figs. 3 and 5, which are 104-piece jigsaw puzzles of overall size 18 in $\times$ 13 in. These are two-dimensional (2-D) rectangular puzzles which are arranged in a grid. Every interior puzzle piece has four neighbors, and every frame piece (except the four corner pieces) has three neighboring pieces. As will be seen later, the grid-like form is not essential for the successful solution of the problem.

The puzzle pieces are presented to the vision system one by one in a random order. The camera is assumed to be at the same height directly above the pieces, so that a piece shape is acquired modulo translation and rotation. Our task is to solve the puzzle, namely, to find the correct configuration of all the pieces, and the relative translation and rotation of each pair of neighboring pieces, so that the actual assembly can be established.

B. Local Matching

In this section we desribe the shape acquisition and the local shape matching. The following procedure is applied:

1) The puzzle pieces are photographed one by one by a black and white RCA 2000 camera from the same distance and viewing angle, and the pictures are digitized and thresholded to get a binary image for each puzzle piece.

2) The boundary of each piece is extracted from the binary image. These are our *experimental* shapes.

3) A *smoothing* procedure is applied to each boundary curve. We use the procedure which is described in detail in [17]. Basically, this expands the noisy boundary curve to a narrow strip defined by a certain threshold value ϵ and then finds the shortest polygonal path lying in this 2ϵ-wide strip. (It may be imagined as stretching a loose rubber band within a narrow sleeve.) This gives a polygonal approximation of each observed curve.

4) The boundary curve of each piece is divided into four subcurves corresponding to the four sides of the puzzle piece and these curves are later used in the matching procedure. This division is based on finding four, so-called *breakpoints* on the boundary curve of each piece. These *breakpoints* are points of sharp convexities on the boundary of the piece (see [17] and [19] for the description of a procedure to find these *breakpoints*).

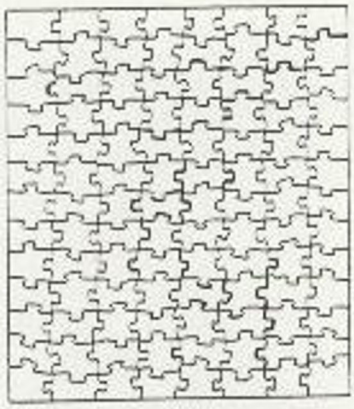


Fig. 3. An assembled jigsaw puzzle.

5) Each curve is sampled at equal arclength and represented by the sequence of (x, y) coordinates at its sample points.

6) Pieces having an (almost) straight section between adjacent corners are identified as *frame* pieces (see Section II-C for the importance of frame piece identification).

7) A local curve-matching procedure is applied. For each pair of two different puzzle piece boundary subcurves (one of four from each piece), the subcurves are matched using a matching algorithm due to Schwartz and Sharir (see [14], [17]), which finds the rotation and translation minimizing the least squares sum of the distances between corresponding sample points on the two subcurves. Specifically, if the two subcurves are represented by the sequences of 2-D points $(u_j)_{j=1}^{n}$ and $(v_j)_{j=1}^{m}$ (let $n \le m$), it finds the Euclidean transformation E of the plane minimizing the l^2 distance

$$\min_{d} \min_{E} \sum_{j=d+1}^{d+n} |Eu_j - v_j|^2, \qquad d = 0, \cdots, m-n.$$

The obtained *best* least squares distance is later used as a matching (penalty) score for the corresponding subcurves in the global matching algorithm.

The actual computation time of the least squares matching score for two curves is less than a second, however, for large puzzles, the pre-computation of all the matching scores is the most time-consuming step of the algorithm. There are a number of ways to speed it up. First, one does not have to compute all the *local* matching scores except those scores which will be actually required by the *global* algorithm (Section II-C). Second, a fast preliminary scoring procedure can be applied to eliminate obviously unsuitable candidates for mating. Such a procedure can be based on features such as convexity/concavity, arclength, etc. (see [12]). Finally, all the *local* matching scores can be computed in parallel.

Remark: The *breakpoints* heuristic (item 4, above) has been used since we applied the local curve matching of [17] which requires one of the matched curves to be a proper subcurve of the other. Recently, this matching algorithm has

been extended to the more general case of two curves with an arbitrary, *a priori* unknown matching portion (see [20]). Hence the *breakpoint* heuristic is unnecessary, if the latter curve-matching algorithm is applied.

C. The Global Puzzle Assembly Algorithm

The global puzzle assembly algorithm consists of two major substeps. The frame of the puzzle is assembled first, and then it is used as a starting point for assembly of the entire puzzle. This is also the common human heuristic in puzzle assembly. However, our consideration to apply it has been purely mathematical. If we view the assembly of a full jigsaw puzzle as a 2-D problem, the frame assembly may be considered as an assembly of a one-dimensional (1-D) circular puzzle. (By a 1-D puzzle we mean a puzzle consisting of only one row (or column) of pieces.) Thus the frame assembly should be inherently easier. Moreover, as will be seen later, it gives a solid starting point to tackle the assembly problem of the entire puzzle.

Frame Assembly: The frame pieces are recognized by their straight line side (see item 6 in Section II-B). Let us name the sides of a piece in the order of their appearance counterclockwise—bottom, right, top, and left side, the straight line being the bottom side (see Fig. 2). The exceptional four corner pieces have two bottom sides and one right and left side. It is easy to see that a correct matching along the outer frame may occur only between a right side of one piece with a left side of another. Let K be the number of frame pieces, and assume that some arbitrary order has been induced on these pieces. Our task is to find a (cyclic) K permutation which will give the correct order of the pieces around the frame.

Define a matrix $M(i, j)$, $i, j = 1, \cdots, K$, by agreeing that $M(i, j)$ should equal the inter-curve L^2 distance obtained by best matching the right side of piece i to the left side of piece j (see item 7 in Section II-B). Any choice of exactly K entries, one from each row and one from each column, defines a K permutation P of the frame pieces so that $P(i) = j_i$, where j_i is the entry chosen from the ith row. Since for the correct piece order the corresponding piece sides should match, their L^2 distance has to be very small (ideally, zero). Hence, we are looking for K entries in M, one in each row and one in each column, such that the sum of these K entries is minimal and the K permutation P is a cycle. (An alternative approach can try to minimize the maximal entry.) Suppose that these entries have been determined to be $j_1, \cdots, j_K$, where j_i is the entry in the ith row. Then we can assemble the outer frame by simply putting piece j_i next to the right of piece i.

The combinatorial problem that we have just stated is well known in the literature as the *traveling salesman problem* (see, for example, [21], [22]) which is usually formulated as follows:

> *Given K cities and their pairwise distances, find the shortest closed path which passes through every city exactly once.*

In our case we must solve the so-called *asymmetric* problem, which means that the distance from city i to city j is not necessarily equal to the distance from city j to city i.

The *traveling salesman* problem is known to be *NP*-complete (see [18]), but for small K's there are efficient algorithms that solve it (see, for example, [22]). (Note that in our example $K = 2 \times (13 + 6) = 38$.) These algorithms may be divided into two main categories:

a) Time-efficient algorithms which use heuristic methods and therefore only assure discovery of the optimal solution with high probability (see, for example, [21], [23]).

b) Algorithms which always find the optimal solution, but in some cases may be very time consuming (see [24]).

We actually use the algorithm of type a) which was published in [25], and is based on a heuristic tour building approach of [23], [26] enhanced by an improvement phase due to Lin [27].

This type of heuristic algorithm is especially suited for robotic interaction. First, it guarantees fast performance. Second, even if it does provide the full correct tour of the *traveling salesman*, due to the heuristics applied, there is a high probability that significant portions of the tour are correct. Hence the following interactive procedure can be applied (see also Fig. 1):

1) Apply the *traveling salesman* algorithm to the problem defined by matrix M.

2) Feed the solution to the robot, and try to assemble the pieces according to the proposed order. For each proposed matching pair, the robotic fine assembly procedure decides whether the specific match is correct, incorrect, or (possibly) undecidable. This information is fed back to the computer.

3) If all the matches are correct, the frame is assembled. If some of the matches are incorrect, change the matrix M according to the information obtained from 2). Specifically, three cases are observed. If a match (i, j) is correct, assign zero to the (i, j) entry of the matrix and assign "almost infinity" to all other entries of row i and column j, since an ideal match has a zero (penalty) score and only one ideal match exists in a given row and column. If a match (i, j) is incorrect, assign "almost infinity" to the entry (i, j), since wrong matches should attain heavy (penalty) scores. If a match (i, j) is undecidable, make no changes in this entry of the matrix, since no additional information is gained by this check. Then go back to 1) with the improved matrix M.

This iterative procedure has the advantage of quickly reducing the "practical dimension" of the *traveling salesman* problem being considered, especially if a big percentage of the suggested matches was correct. We believe, that a very small number of such iterations will be needed in order to get the correct solution.

We intend to expand our robotic assembly procedure which is described in Section III to accommodate for such an integrative solution. At this stage, our experiments were based purely on computer vision equipment, so no feedback was available, nevertheless we have obtained the correct frame assembly after the first application of the *traveling salesman* algorithm on the original matrix (see Fig. 4). Although the *traveling salesman* is known to be difficult, this result is not entirely surprising, since the solution algorithms are usually evaluated for arbitrary random matrices. In our case, however, the matrix M is not arbitrary, but strongly favorable to

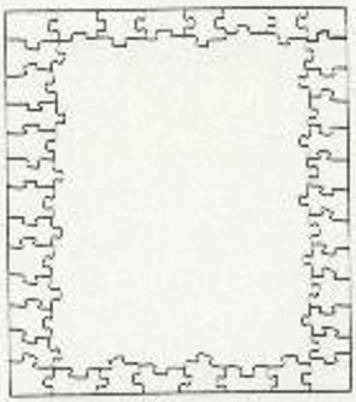


Fig. 4. An assembled frame of the puzzle in Fig. 3.

its optimal solution by virtue of the good performance of the local matching algorithm. In our experiments, the algorithm converged to the correct solution in less than a minute.

The above mentioned strategy will produce a unique solution when the puzzle piece sides have different shapes. This was the case in our experiments, although the shapes of the pieces are quite similar and cannot be visually easily distinguished. The robotic verification step assures us that only "correct" matings are accepted. However, in case there are pieces with identical left and right sides there might be several consistent solutions. In such case one might try and find all these solutions, and reject the wrong ones in the later puzzle interior assembly stage. Also, a number of heuristics can be applied to reject the wrong solutions already in this first stage. One is to check, if the outer frame is actually closed. If not, the proposed solution can be rejected and the iterative algorithm redirected to another solution by exchanging a pair of pieces having a left (right) side with an identical shape. This procedure can be reiterated. In case the puzzle is rectangular, as in our examples, the special four corner piece information can be exploited to constrain the proposed solutions. It should be noted that in our algorithm we did not use this information or the fact that the puzzles are rectangular, except for the identification of the frame pieces.

Arrangement of the Puzzle Interior: After complete arrangement of the outer frame (see Fig. 4), we proceed to the arrangement of the interior pieces. Basically, this is done by considering the four interior corners of the frame (see Fig. 4) and using the fact that an interior piece, which fits into one of the four corners of the frame has to match *two* sides with two previously known frame pieces. This is advantageous because the matching curve is approximately twice as long compared to the previous case.

A simple approach to solve the interior is to use a greedy algorithm. Using such an approach we could start with the piece which has the best matching score with one of the corners. After such a piece is discovered and located, it will create two other corners with the same property. Then, we could look for another piece having the best possible match with one of the remaining corners, and proceed iteratively in this way. Of course, at some stage we will get places in which a piece must match along three of its sides; this strengthens our scoring procedure even more. Finally, the last piece will have to match along four of its sides.

However, in our situation, in which the boundaries of the pieces are quite similar, even when two sides are being matched (see Fig. 2) a greedy algorithm is unlikely to succeed. One way to overcome this problem is by using robotic feedback to verify the proposed matches in every corner. Such an approach can lead to the correct solution, however it will be time-consuming. In this stage, as in the previous one, we prefer to use robotic feedback for verification of partial solutions involving a big number of puzzle pieces, and not at each *local* stage. Hence, as in the frame assembly, we involve first a *global* computer algorithm, to achieve a proposed solution. The algorithm is based on the following backtracking technique, which processes sequentially the puzzle interior corners, beginning with the lower left side of the puzzle interior and advancing within each row to the right. (This procedure is less general than picking up the best corners wherever they may be, but it is easier to program and the loss of information is nonessential.)

1) For the first corner, all the puzzle pieces, which are not frame pieces, are matched at this corner and their local matching score is computed. The results are sorted and a prescribed number of best solutions (denoted as KBEST) is passed to the next stage.

2) At the second corner the same procedure is repeated for each of the KBEST partial solutions which passed the previous stage, and only KBEST *overall* (for both executed steps) best solutions are passed to the next stage.

3) The algorithm then proceeds iteratively. At the last corner in each row we have three sides to match. In the last row we have three matching sides for every piece and four matching sides for the last piece.

In such a way we explore the interpretation tree of the interior puzzle arrangements in an efficient way by pruning all but KBEST solutions at each stage. Of course, the number KBEST need not be the same for all corners but it is more appropriate to vary it from stage to stage. A more sophisticated approach would be to make a dynamic decision as to which solutions should pass to the next stage by assigning an upper bound (as a function of the stage) for the overall matching of a partial solution and to pass along only those solutions that do not exceed this bound.

In our vision experiments (Figs. 3 and 5) we used KBEST = 200, which was kept the same at every stage. Actually, the correct solution in the puzzle of Fig. 3, for example, always lay among the ten best solutions. The running time of the puzzle interior assembly with KBEST = 200 was about 10 min on a VAX 780. The program was written in Fortran and no special attempt was made to optimize the code. The interior assembly algorithm accepted as input the matrix of the *local* matching scores.

Since the solution of the puzzle interior is also based on a heuristic algorithm, an integrated solution, which includes robotic verification, can ensure convergence to the correct solution. Here again, as in the frame assembly case, we can check by a robot the proposed solution after it has been completed, discover its correct and obviously wrong subsets, and return to the computer algorithm with this additional information for another (probably much more successful)

Fig. 5. An assembled jigsaw puzzle.

iteration. This interactive procedure can even be handled dynamically. For example, if at some stage of the puzzle interior assembly all the matching scores become obviously bad (above a certain dynamic threshold), it indicates that we are heading towards a wrong solution, since, probably, the correct solution has been pruned at some earlier stage. This is an obvious place to halt the backtracking algorithm and check the best partial solutions achieved so far by robotic assembly. After some of the matchings are verified, and others rejected, the backtracking algorithm can proceed from a more solid starting point.

In our visual experiments we have also developed an efficient algorithm for simultaneous assembly of a number of jigsaw puzzles and successfully assembled the 208 pieces of the puzzles in Figs. 3 and 5 into two different puzzles. The reader is referred to [14] for the details of this method, as well as for the discussion on the complexity of the above mentioned algorithms.

III. Robot Assembly of Puzzle Pieces

Any physical assembly process raises problems related to part availability and detection, retrieval and stable grasping, friction, and jamming. Position, sensing, and control errors make assembly success uncertain. Additionally, the robot work envelope is reduced by the jigs and fixtures that keep parts in place. When industrial assembly is attempted, feeders, conveyor belts, and fixtures are standard. The specifics of jigsaw puzzle shape makes this solution impractical, since it is hard to design a feeder that can accommodate all the varying puzzle pieces. A large number of jigs would also drastically limit the robot work envelope. Moreover, grasping becomes a problem when using a parallel gripper on the curved shapes characteristic to jigsaw puzzle pieces. In view of these assembly difficulties, we designed an installation that aims to assure uniform and stable grasping, maximum robot workspace, and good sensor feedback.

A. Experimental Installation

This section presents an experimental installation used in the puzzle assembly, integrating a vision system, robot controller, and an additional master computer. Subsequently, we detail the modifications made on an off-the-shelf wooden puzzle to assure a stable robot grasping and eliminate the need for jigs and fixtures.

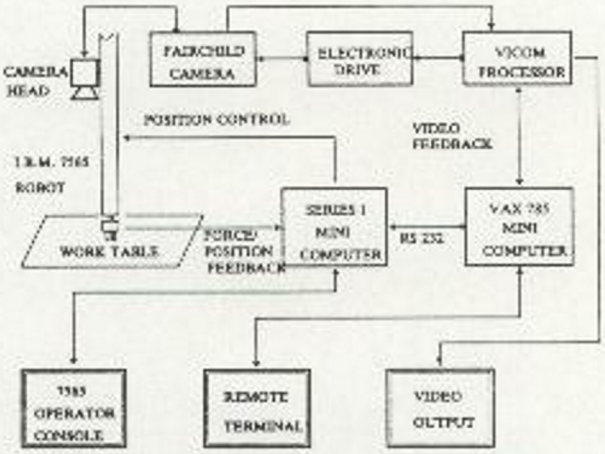


Fig. 6. Experimental installation for puzzle assembly.

Puzzle assembly is a computationally intensive process that incorporates visual data acquisition and procesing, routines for piece matching, as well as robotic assembly routines. It became evident that the volume of computation involved exceeds the capacity of the IBM 7565's Series 1 robot controller used for assembly. Thus we used a more powerful Vax 785 computer to do the heavy computations related to vision and planning, as presented in Fig. 6. Local force data analysis done by the robot controller then permits real-time control, which would be impossible under remote control.

The vision system uses a Fairchild CCD camera and a VICOM image processor with a 512 × 512 array. The Fairchild camera sensor head was installed directly on the robot arm, at a fixed height above the robot work table. This eliminates the need for special lens drives to compensate for variable focal lengths, while allowing a good scan of the table. For our experiments we selected an off-the-shelf wooden puzzle with tolerances of 0.04–0.06 of an inch. Our visual matching algorithm uses only the piece shape, therefore we covered all piece pictures with black nonreflective paper. It is important that the pieces do not slip on the metal work table during robot assembly. Therefore, the bottom of the puzzle pieces was fitted with Koroseal flexible magnetic strip [28] to prevent sliding at small assembly forces. The magnetic force is limited by piece size to about 4 N for a 2- to 3-in^2 area. The force that the robot uses during assembly has to be of the same order of magnitude so that stationary pieces do not slip on the table top. A cylindrical handle was mounted on each puzzle piece at its center of gravity to assure uniform grasping. Specially designed nails were added to the gripper to increase finger-handle contact surface and amplify assembly forces for better force sensing by the gripper fingers, as shown in Fig. 7.

B. Robotic Assembly of Two Puzzle Pieces

This section describes the strategy used to integrate the vision algorithm with intelligent, force-feedback-based, assembly planner in order to physically interlock two 3-D puzzle pieces. Although the vision algorithm does not require a special puzzle shape in order to match boundaries, the robotic assembly actively exploits the generic shape of the puzzle pieces in Fig. 2. We use the fact that each interlocking side has either a convexity *or* a concavity, but not both. Also exploited

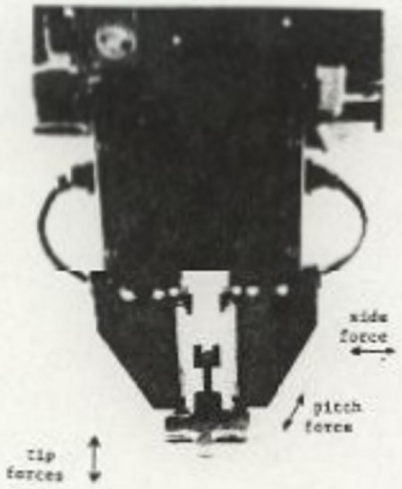


Fig. 7. Robot gripper with grasped puzzle piece.

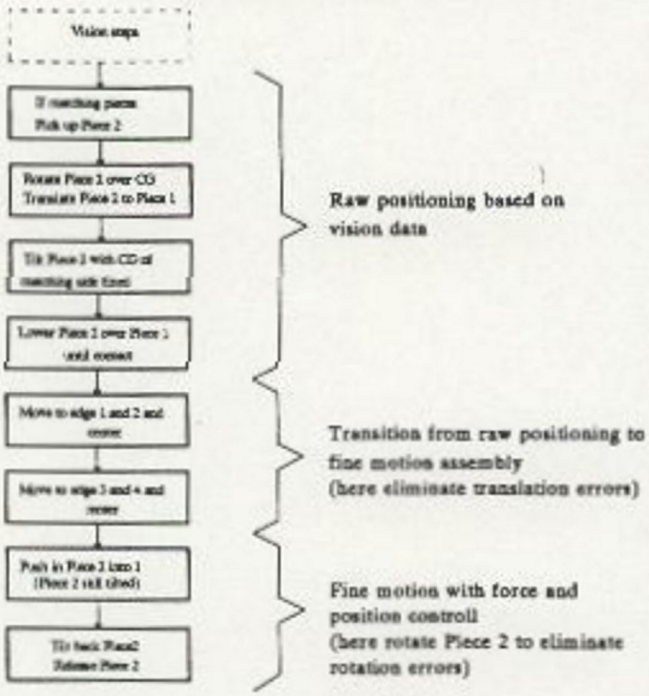


Fig. 8. Robot assembly steps.

is the presence of two low-curvature segments on each matching side (subsequently called "elbows"). Therefore, the procedures described in this section may not be directly extended to a different type of puzzle.

The physical assembly steps are shown in Fig. 8. We named the stationary piece *Piece 1*, while *Piece 2* is the one manipulated by the robot. These steps become building blocks for the assembly of multiple pieces as described in the subsequent section.

1) Vision Steps for Assembly: The puzzle pieces are placed randomly on the work table within the robot work space and within the camera field of view. In this way, the pieces are both visible and reachable. This scanning space is subdivided in quadrants with overlapping borders. During this initial assembly stage, the algorithm determines whether a puzzle piece exists in a given quadrant. If a piece exists, then its raw center of gravity in vision coordinates is determined. In

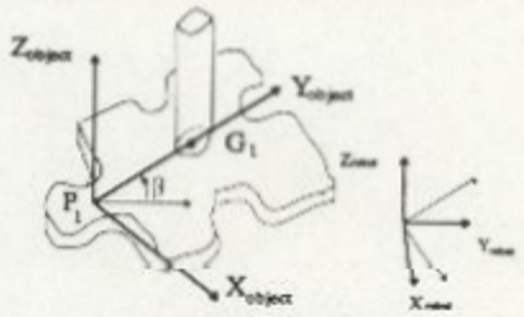


Fig. 9. Object system of coordinates.

general, the piece center of gravity does not coincide with the center of an image. In an extreme case only part of the piece shape is visible to the camera. The robot has therefore the additional task of centering the camera on top of a discovered piece in order to minimize vision errors. At the same time, the assembly planner samples robot arm coordinates when a piece is centered in the picture. This is used later to give a goal for raw assembly steps. The visual search and piece pickup are based on the correspondence between camera field of view and robot work space. This relation is determined by the camera position offset, x/y distortion factor, and projection factor.

Once the vision algorithm has acquired a piece, a new search is initiated for a subsequent piece. The *local* piece matching procedure (see Section II-B) is applied, and the vision algorithm returns the relative rotation and translation of the matching sides. This rotation and translation is then transformed into the corresponding motion of the handle on the puzzle piece. This motion goal is then sent to the assembly routines for execution.

2) Robot Assembly Steps: Simunovich [29] states that the robot range of motion for gross assembly must be about two orders of magnitude greater than the characteristic size of parts to be assembled, while accuracies required for gross assembly are of the same order of magnitude as the size of the parts. He describes fine motion assembly as having a range of the same order of magnitude as the part characteristic size. Accuracies may be as much as three orders of magnitude smaller than the part size. For puzzles with a geometry similar to that presented in Fig. 2, the characteristic size may be the radius of the concavity of the matching side. From this perspective, the automated assembly of two pieces may be structured in three steps: *raw positioning, transition to fine assembly,* and *fine assembly*. During raw positioning, Piece 2 is picked up, rotated and translated, then tilted and lowered on top of Piece 1. When determining motion trajectories for assembled parts it is necessary to attach a system of coordinates to each part being manipulated. The specific geometry of a given part determines the position and orientation of the object system of coordinates attached to it. In the case of a puzzle piece, we define a Cartesian object system of coordinates with the origin at the center of gravity of the matching side P_1. Y_{obj} axis passes through the center of gravity of the piece G_1, as shown in Fig. 9. For square puzzles such as the ones we used, this divides the piece shape into a quasi-symmetrical area with respect to P_1G_1. The choice of P_1 as origin for the object system of coordinates is related to the vision solution which is

given in terms of P_1. While the centers of gravity of two matching sides will not exactly coincide, tolerances are such that they can be overcome during the assembly process. The fact that tolerances are not very small combined with the relatively small thickness of the puzzle pieces reduces the chance of wedging. Friction between parts is therefore minimal during assembly.

The robot gripper fingers have tip, side, and pitch strain gauges giving force readings in three directions as shown in Fig. 7. Readings on the gripper sensors are influenced by a number of factors such as orientation, sensor offset, calibration, mechanical amplification, crosstalk, vibrations. Tests have shown [30] that small rotation errors do not influence force readings substantially. However, a large error (135°) may return readings that differ from actual forces by as much as 100 percent. Therefore, force readings during assembly might depend on gripper orientation during part pick-up. It is therefore important to determine what is the best gripper grasping position in order to maximize sensor output during assembly moves. We consider this to be perpendicular to the general direction P_1G_1, since this is the direction in which initial contact forces appear. After Piece 2 has been grasped and tilted, the raw positioning task is to bring Piece 2 in contact with Piece 1. Inoue [11], Lozano-Perez [4], and others have shown that a purposely induced positional error with a magnitude larger than the uncertainty will disambiguate the configuration of assembled pieces. In the case of a circular peg this induced error results in the peg being lowered aside rather than into its matching hole. From this known configuration a sliding motion with monitored contact forces will find the hole. The same principle is applied in our strategy which induces an overshoot of Piece 2 past the edge of Piece 1. The last move during raw positioning assembly is a vertical move that brings the pieces in contact as shown in Fig. 10(a). This is a force-feedback-based *guarded move* that terminates upon contact with Piece 1. Salisbury [31] gives the relation between position errors and the resulting contact forces in the generalized spring equation

$$f_s = k(x - x_c) \tag{1}$$

where

- x actual robot position
- x_c commanded robot position
- f_s sensed reaction force
- k system compliance constant.

Hence, due to the compliance of the robot arm and puzzle piece, there is a linear relation between the magnitude of positional errors and the resulting contact forces. This is exploited in the setting of force thresholds for the fine assembly guarded moves. No active servo force feedback is applied.

During *transition moves*, the two pieces remain in contact while their relative position changes to a nested configuration. It is during transition moves that translational errors are eliminated. The first transition move is a slide on top of Piece 1 that terminates when contact is lost at edge 1, as presented in Fig. 10(b). The sliding proceeds along the generalized direction P_1G_1. A fundamental principle in assembly is that

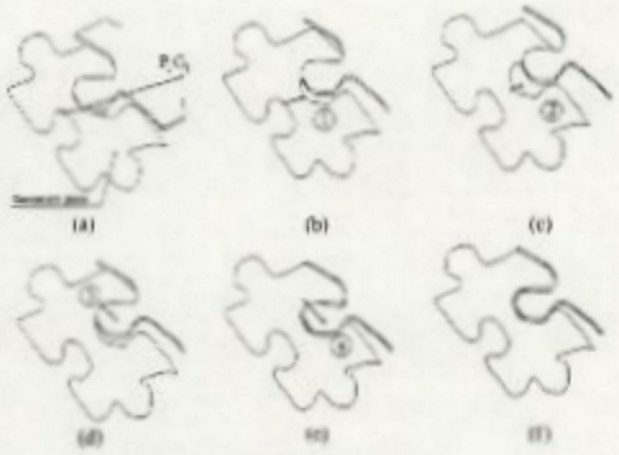


Fig. 10. Transition moves.

the tilting of one of the parts reduces the initial contact area, which means that tolerances become larger. When executing the move in Fig. 10(b), Piece 2 is tilted and larger tolerances increase the chance to successfully intersect edge 1 of Piece 1. After edge 1 was detected, the next move is towards edge 2, where contact between the two pieces is reestablished as in Fig. 10(c).

An average of the two termination points represents the target for the next move. The same procedure is repeated for edges 3 and 4, but the move proceeds along a trajectory perpendicular to P_1G_1, as in Fig. 10(d) and (e). Averaging the four contact points eliminates translation errors introduced by the vision system (Fig. 10(f)).

The final and crucial step is *fine assembly* based on force feedback. During fine assembly, Piece 2 must be pushed down into Piece 1 and then tilted back to a horizontal position. The tilt back must maintain contact with Piece 1 and rotate vertically around the contact point.

While transition moves eliminate gross translational errors, push-in moves have to succeed despite rotational errors. First, the convexity of Piece 2 is brought in contact with Piece 1 by moving Piece 2 back to edge 1 along G_1P_1. Once the contact is established, the next move pushes Piece 2 into Piece 1. The tilting angle θ is such that if rotation errors exist, the *elbow* of Piece 2 hits the top of Piece 1 before the pushing in completes. The algorithm distinguishes between the two possible cases of *right or left elbow hit*. The force-torque system that corresponds to these cases is presented in Fig. 11. During the last move of fine assembly Piece 2 is brought back to a horizontal configuration as shown in Fig. 12.

Force readings during assembly are presented in Fig. 13. These readings represent processed data that have been compensated for sensor offset and calibration amplification. Measured force envelope during piece nesting is of the order of 5 N, or 2.5 N for 50-percent mechanical amplification due to handle-gripper combination. This force level meets conditions set earlier regarding magnetic retention forces, and no slip is present during initial assembly steps. Dynamic effects have not been considered here since assembly is done at low speeds (0.2 in/s). This is necessary in order to improve system response to guarded move termination flags. During fine assembly steps forces occasionally pass the limit of magnetic pull resistance, and slip is present.

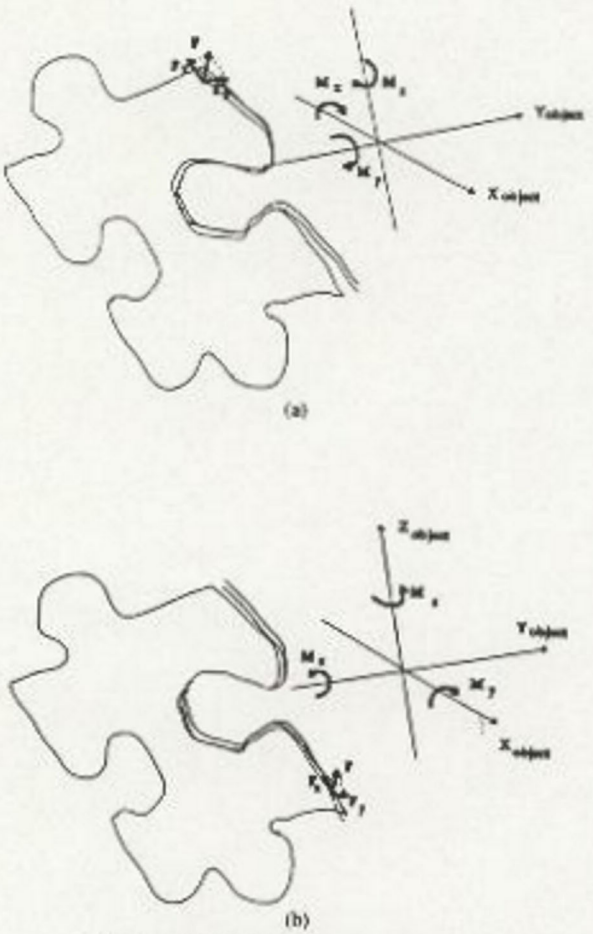


Fig. 11. Force–torque system for elbow hit.

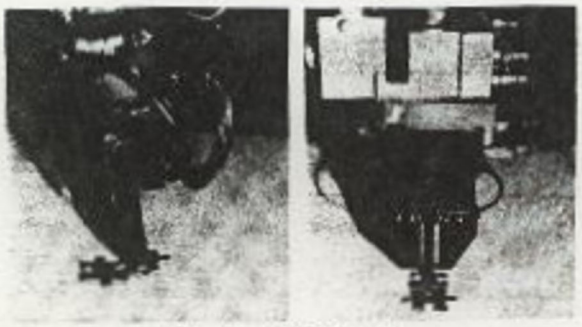
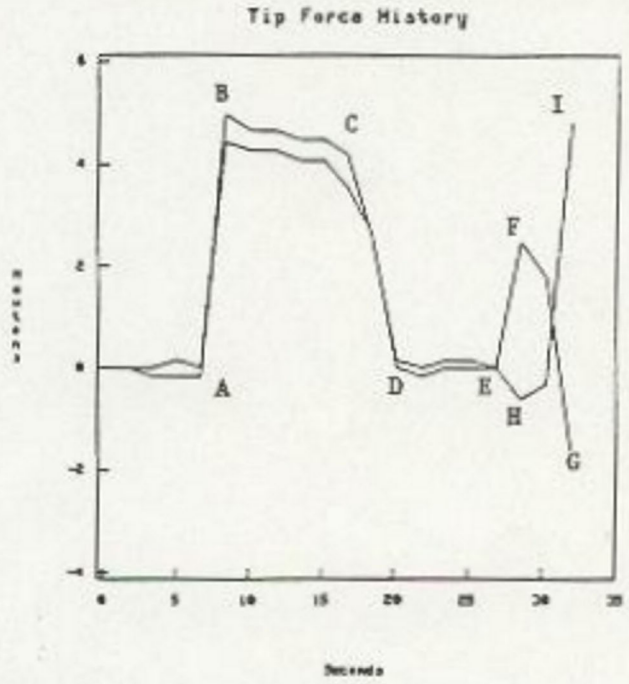


Fig. 12. Tilt-back move.

When analyzing Fig. 13 we note a large tip force increase from A to B, followed by a sharp decrease C to D. These represent the end of raw positioning moves at point B and the detection of edge 1 at point D, as described previously. Both left and right finger sensor outputs follow the same pattern $ABCDE$, but separate along EFG and EHI, respectively. This corresponds to nesting moves that detect edges 3 and 4, when sensors are subject to torques in opposing direction.

To test the algorithm robustness it is important to estimate the limit on rotational and translational errors that the assembly algorithm can tolerate. We define the vision rotation error θ_{err} as

$$\theta_{err} = \theta' - \theta \tag{2}$$



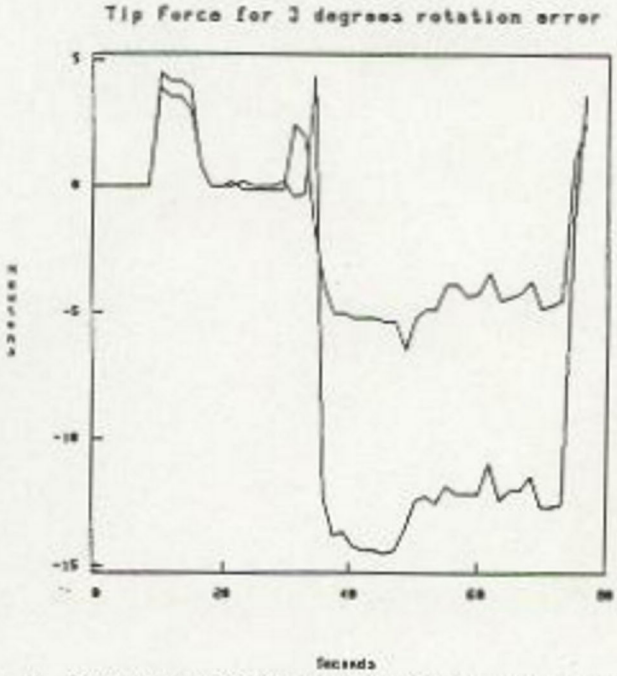


Fig. 13. Force readings during assembly (total tip force for left and right sensors).

where

- θ_{err} rotation error from vision solution
- θ' solution returned by vision match
- θ correct solution.

Due to puzzle asymmetry, any rotation errors produce translation errors of the initial contact point. This translation errors are given by

$$X_{1'} - X_1 = G_{1x\,\text{robot}'} - G_{1x\,\text{robot}} + 2(T+l) \cdot \sin\left(\frac{\theta_{err}}{2}\right) \cos\left(\frac{2\beta + \theta_{err}}{2}\right) \tag{3}$$

$$Y_{f'} - Y_f = G_{3y\,\mathrm{robot}'} - G_{1y\,\mathrm{robot}} + 2(T+l) \cdot \sin\left(\frac{\theta_{err}}{2}\right) \sin\left(\frac{2\beta + \theta_{err}}{2}\right) \quad (4)$$

where

- $X, Y_{f'}$ robot coordinates of contact point with vision rotation error
- X, Y_f robot coordinates of contact point without rotation errors
- θ_{err} rotation error from vision
- T $= a - \cos(\theta)a + (b + f)\sin(\theta)$
- a, b, f distances as described in Fig. 19 (Appendix).

The translation error from vision may not be larger than the pseudo radius of the stationary puzzle concavity, or the peg will not intersect it. This is due to Piece 2 tilting which results in a small contact surface at the peg edge, rather than contact on the total peg surface. This can be seen in Fig. 10(a). Another condition requires the gripper to be as close as possible to the perpendicular to P_1G_1, in order to assure symmetry in sensor readings. For a pseudo radius of 0.25 in, the condition set in (3) returns a maximum θ_{err} of 4° for successful assembly. Tests have been run with rotation errors varying between 0° and 4.6°, with successful nesting at 3° and failure at 4.6°. In order to improve assembly robustness it may be necessary to modify the nesting algorithm to take alternative actions in case edge 1 was not detected. The algorithm might reverse the scan on a perpendicular direction to P_1G_1 to detect edges 3 and 4 and *then* detect edges 1 and 2. Finally, assembly success also depends on the friction between the two pieces, as well as on the force sensor sensitivity. For example, the horizontal friction force at the contact between Piece 2 and Piece 1 at the first transition move should be less than the magnetic force that keeps Piece 1 stationary. Otherwise, Piece 1 will be dragged along on the work table. The limit on friction between the two pieces requires high sensor sensitivity as well as low system inertia. Such a system is necessarily slow which results in long overall assembly time. Robot assembly time was about 70 to 80 s excluding the time allocated to the vision algorithm.

From the above discussion we see that the restriction imposed by the assembly strategy on the geometry of the assembled pieces is the existence of a concavity or a convexity on the matching side (but not both). A second restriction is the existence of two segments of small curvature (previously called "elbows"), with lengths of the same order of magnitude as the diameter of the concavity. The existence of one concavity assures initial elimination of translation errors versus the center of gravity of the matching side, provided that these errors are within the bounds previously described. In this situation the interior of the concavity is found by the tilted and moving part. The existence of the "elbows" combined with small tilting angles result in contact forces which occur before the push-in motion completes. This represents a way to detect rotational errors. It is expected that the same strategy will apply to the mating of other parts with similar shapes, such as gear-pump shafts.

C. Assembly of Multiple Puzzle Pieces

Robot assembly of multiple dissimilar industrial parts has not yet been intensively studied. Difficulties associated with this task are due to the changing geometry of assembled pieces. No single strategy, however efficient, will succeed in dealing with many, different shapes. Tolerances with which each part is manufactured represent an additional problem. These tolerances induce uncertainties that accumulate as the assembly progresses. Large uncertainties may rend assembly of later pieces impossible. In general, multiple part assembly [32] may involve either different operations applied to dissimilar parts, or a common operation applied repeatedly to several similar sets of parts (e.g., nut–washer–screw assembly). When several parts are assembled, it is essential to limit error propagation, in order to solve the task.

This section addresses the problem of four inner puzzle-piece assembly, and requires modeling uncertainty propagation in a four-piece kinematic chain. This model bounds accumulated uncertainties that have to be dealt with during assembly of the fourth piece, when that piece must interlock with two prior pieces. The technique presented may be extended to complete assembly of multi-piece puzzles.

Brooks [33] and Smith and Cheeseman [34] give methods for error analysis and uncertainty estimation, by using compounding and merging of approximate transformations to obtain a basic estimation of uncertainties at the end of a series of operations. At each intermediate step an additional uncertainty specific to that step is added. If no remedial action is taken, then a large error may result between the desired and actual final positions. In the case of puzzle assembly, the tolerances with which pieces are formed allow for a small degree of relative motion even after they are assembled. Each piece can both rotate and translate relative to its neighboring piece. Their matching side may then be viewed as a three-degree-of-freedom link (see Fig. 20 in the Appendix). Piece 4 has to interlock on two sides, rather than on one side only, as do Piece 2 and Piece 3. Depending on the magnitude of rotational and translational errors, uncertainty propagation may make the assembly of Piece 4 impossible, as shown in Fig. 14. A strategy that succeeds despite previously described assembly uncertainties is presented in Fig. 15. This strategy is based on the principles of limiting uncertainty growth and inducing errors to disambiguate piece position during assembly. When locking Piece 2 into Piece 1 the key move is bringing Piece 2 in the configuration presented in Fig. 16(a). This step uses (see [4], [11]) induced gross position errors to disambiguate piece orientation. Force feedback stops the motion of Piece 2 when corner A makes contact with Piece 1. The assembly of Piece 4 may be viewed as having two phases. In the first phase, Piece 4 is tilted and assembled in Piece 2, using the methods outlined in Section II-B. At the end of this phase, Piece 4 has eliminated translational and rotational errors versus Piece 2, and is still tilted versus the work table, as shown in Fig. 17. The next step is to tilt back Piece 4 so that it locks in Piece 3. Due to the rotation of Piece 2, Piece 4 hits the "elbow" of Piece 3 in a known orientation. Finally, Piece 2 is pushed away until the contact between Piece 4 and Piece 3 is lost, and the tilt-back move completes.

Fig. 14. Uncertainty effects on four puzzle piece assembly.

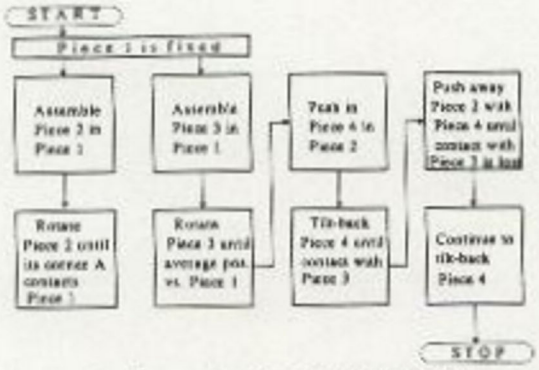
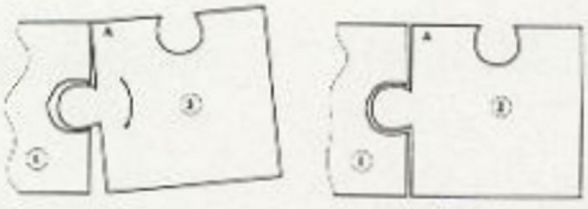


Fig. 15. Strategy for four puzzle piece assembly.



(a) (b)

Fig. 16. Strategy for Piece 2. (a) Initial configuration. (b) Final configuration.

The strategy previously discussed necessitates pushing Piece 2 away from Piece 3, while tilting back Piece 4 towards its final horizontal configuration. During this final assembly step, Piece 2 slides over the work table despite magnetic friction.

During sliding motion, the vertical component of the pushing force is negligible, so 2-D analysis of sliding objects is applicable [1], [35]–[39]. Mason and Peshkin have shown

Fig. 17. Assembly configuration before tilting back Piece 4.

that a sliding object has three degrees of freedom, one rotation and two translations. When the object is pushed, one degree of freedom is lost due to interaction with the pushing object. In his analysis, Peshkin uses a configuration map that describes simultaneously the geometric and physical consequence of the pushing operation on *all* initial configurations. This configuration map is then used for planning a series of pushing motions designed to bring the object in a desired position and orientation.

The case of a pushed puzzle piece is somewhat simpler. When an *assembled* puzzle piece is pushed one more degree of freedom is lost. The piece is free to rotate around a center of rotation (COR) constrained in the concavity of the matching side, as shown in Fig. 18. The distance from COR to the piece center of gravity (CM) x_r is known based on the piece geometry. Force F is the contact force necessary to push Piece 2 despite friction with the robot table, assuming Coulomb friction and quasi-static equilibrium.

The strategy previously described calls for sliding of Piece 2 combined with tilting Piece 4 towards its final horizontal configuration. Rather than executing pushing and tilt-back moves simultaneously, we chose to divide them into a series of alternating, guarded, tilt-back and pushing moves. In this way it is easier to monitor only one type of force sensor at a time, namely tip forces for tilt-back moves and side forces for sliding moves. Similarly, force thresholds for these guarded moves may be divided into smaller steps, so that forces increase gradually. The division into small moves minimizes the effect of overshoot due to system inertia, and allows for closer force control at the expense of assembly time. At each tilt-back step the goal is computed based on current position, thus gradually eliminating the effect of induced error in the position of Piece 2.

IV. Conclusions and Future Research

We have suggested an algorithm for assembly of large jigsaw puzzles by integration of curve-matching techniques from Computer Vision, combinatorial optimization algorithms, actual robotic verification, and fine assembly. We also presented an implementation of the vision algorithm, and a small-scale robotic assembly. The method should be applicable in any assembly tasks where such thin shapes have to be interlocked. Closed-chain assembly problems could also benefit from the present algorithm.

Several things can be explored in the future, among them:

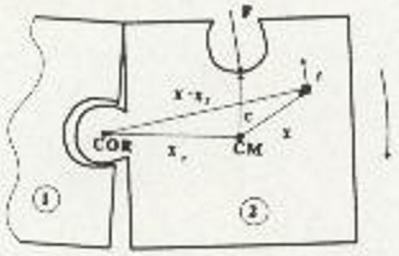


Fig. 18. Sliding forces.

1) Application of more sophisticated curve-matching algorithms (see [40]), which will enable solution of arbitrary shape puzzles.

2) Extension of the present assembly module into a module capable of dealing with large jigsaw puzzles as suggested in Section II-C.

3) Use of dextrous manipulation instead of parallel fingers. This will eliminate the need to attach a handle at the center of gravity of the puzzle pieces. Also, we can benefit from the "rich" force data at each finger, giving more information for the verification step.

4) Study of encreased rotation error effects on force sensor readings to improve algorithmic robustness.

Appendix

When two pieces are interlocked the push-in motion has goals given by (A1)–(A5) (see Fig. 19).

$$l = \frac{Z_{\text{table}} - (f + b + t)\cos(\theta) - t}{\sin(\theta)} \tag{A1}$$

where

- l distance from the contact point to Piece 2 center of gravity
- t piece thickness
- θ tilting angle
- β angle between Y_{robot} and Y_{object} as shown in Fig. 9
- a distance P_1G_1

$$X_{\text{pushin}} = X_{\text{robot}} - t\sin(\beta)\sin(\theta) \tag{A2}$$

$$Y_{\text{pushin}} = Y_{\text{robot}} + t\cos(\beta)\sin(\theta) \tag{A3}$$

$$Z_{\text{pushin}} = Z_{\text{robot}} - t\cos(\theta) \tag{A4}$$

$$\text{Pitch} = \theta. \tag{A5}$$

When multiple pieces are assembled, a Cartesian system of coordinates X_iY_i is attached to puzzle Piece i as described in Section III-B2. A second Cartesian system of coordinates $X_{i'}Y_{i'}$ corresponds to the average position that *Piece i* may take once it was assembled in Piece $i-1$.

Small assembly uncertainties may be expressed by triplets $(\delta X_{ii'}, \delta Y_{ii'}, \delta\theta_{ii'})$. Each new assembled piece adds to the uncertainty so that at the end of the kinematic chain, P deviates from the goal G. Thus it is necessary to determine the range of possible positions of $P(x_p, y_p)$ in a fixed system of coordinates X_0Y_0. We chose to attach this system to puzzle *Piece 1* and consider it fixed during the whole assembly. P coordinates in

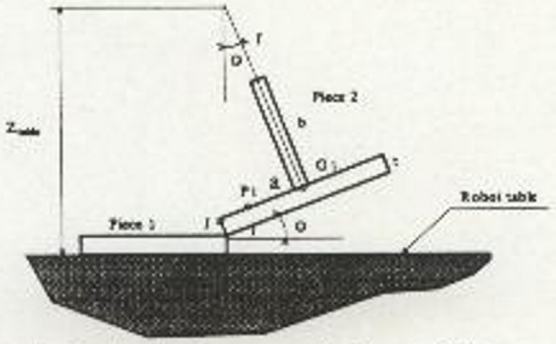
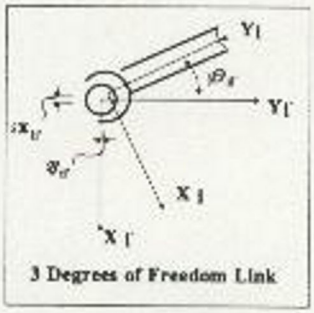


Fig. 19. Notation for contact point distance calculation.



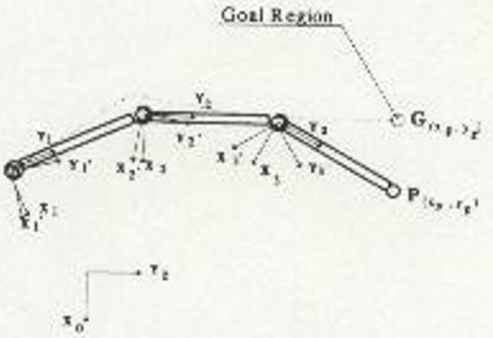


Fig. 20. Uncertainty propagation model.

the system of coordinates X_0Y_0 may be found with (A6)–(A8). A measure of the final uncertainty is then given by $\sqrt{(x_g - x_p)^2 + (y_g - y_p)^2}$.

$${}^0P = {}^0_1T'\,{}^1_1T\,{}^1_2T'\,{}^2_2T\,{}^2_3T'\,{}^3_3T\,{}^3P \tag{A6}$$

where ${}^0P$ are P's coordinates in the fixed system X_0Y_0; ${}^i_i T'$ is the transformation between the uncertain system of coordinates i and the correct one i'; ${}^3P$ are P's coordinates in the uncertain system X_3Y_3 attached to the last puzzle piece in the kinematic chain (see Fig. 20).

The matrix transformation ${}^i_i T$ may be written as

$${}^i_i T = \begin{bmatrix} \cos\theta_{ii} & -\sin\theta_{ii} & 0 & \Delta x_{ii} \\ \sin\theta_{ii} & \cos\theta_{ii} & 0 & \Delta y_{ii} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{A7}$$

${}^i_i T'$ are small rotations and translations and may be written as

$${}^i_i T' = \begin{bmatrix} 1 & -\epsilon_{ii'} & 0 & \delta x_{ii'} \\ \epsilon_{ii'} & 1 & 0 & \delta y_{ii'} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{A8}$$

REFERENCES

[1] M. T. Mason, "Manipulator grasping and pushing operations," Ph.D. dissertation, MIT, Cambridge, MA, 1982.

[2] ——, "Compliance and force control for computer controlled manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, 1981.

[3] S. J. Buckley, "Planning and teaching compliant motion strategies," Ph.D. dissertation, MIT, Cambridge, MA, Jan. 1987.

[4] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *Int. J. Robotics Res.*, vol. 3, no. 1, pp. 3–24, 1984.

[5] T. Lozano-Perez, "Task planning," in *Robot Motion Planning and Control*, M. Brady *et al.*, Eds. Cambridge, MA: MIT Press, 1984, pp. 473–498.

[6] D. E. Whitney, "Force feedback control of manipulator fine motions," *J. Dyn. Syst.*, Meas. Contr., pp. 91–97, June 1977.

[7] ——, "Part mating theory for compliant parts," Tech. Rep. R-1407, The Charles Stark Draper Lab., Inc., Cambridge, MA, 1980.

[8] M. S. Ohwovoriole and B. Roth, "A theory of parts mating for assembly automation," Tech. Rep., Stanford Univ., Stanford, CA, 1981.

[9] M. Erdmann, "Using backprojections for fine motion planning with uncertainty," *Int. J. Robotics Res.*, vol. 5, pp. 19–44, 1986.

[10] ——, "On motion planning with uncertainty," Tech. Rep. AI-TR-810, MIT, Cambridge, MA, 1984.

[11] H. Inoue, "Force feedback in precise assembly tasks," Tech. Rep. AI Memo 308, MIT, Cambridge, MA, 1974.

[12] H. Freeman and L. Garder, "Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition," *IEEE Trans. Electronic Comp.*, vol. EC-13, pp. 118–127, Apr. 1964.

[13] G. C. Burdea, "Robot assembly of jigsaw puzzle pieces," Ph.D. dissertation, Computer Science Dept. Courant Inst. of Math., NYU, New York, NY, 1987.

[14] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan, "Solving jigsaw puzzles by computer," *Annals Oper. Res.*, vol. 12, pp. 51–64, 1988.

[15] G. M. Radack and N. I. Badler, "Jigsaw puzzle matching using a boundary centered polar encoding," *Comput. Graph. Image Process.*, vol. 19, pp. 1–17, 1982.

[16] G. Burdea and H. Wolfson, "Automated assembly of a jig-saw puzzle using the IBM 7565 Robot," RR-55, Robotics Lab, Courant Inst. of Math., NYU, New York, NY, 1985.

[17] J. T. Schwartz and M. Sharir, "Identification of partially obscured objects in two dimensions by matching of noisy "Characteristic Curves"," *Int. J. Robotics Res.*, vol. 6, no. 2, pp. 29–44, 1987.

[18] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., 1979.

[19] A. Kalvin, E. Schonberg, J. T. Schwartz, and M. Sharir, "Two dimensional model based boundary matching using footprints," *Int. J. Robotics Res.*, vol. 5, no. 4, pp. 38–55, 1986.

[20] H. Wolfson, "On curve matching," in *Proc. IEEE Computer Society Workshop on Computer Vision* (Miami Beach, FL, Nov. 1987), pp. 307–310.

[21] N. Christofides, *Graph Theory*. New York, NY: Academic Press, 1975.

[22] E. L. Lawler, J. R. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. New York, NY: Wiley, 1985.

[23] T. C. Raymond, "Heuristic algorithm for the Traveling-Salesman Problem," *IBM J. Res. Develop.*, vol. 13, no. 4, pp. 400–407, 1969.

[24] M. Belmore and J. C. Malone, "Pathology of Traveling-Salesman Subtour-Elimination Algorithms," *Oper. Res.*, vol. 19, pp. 278–307, 1971.

[25] Z. Fenel, "Routing problem," CACM Algorithm 456.

[26] R. L. Karg and G. L. Thompson, "A heuristic approach to solving Traveling Salesman Problem," *Manag. Sci.*, vol. 10, no. 2, pp. 225–248, 1964.

[27] S. Lin, "Computer solutions of the Traveling Salesman Problem," *Bell Syst. Tech. J.*, vol. 44, pp. 2245–2269, Dec. 1965.

[28] B F Goodrich Co., *Korasteel Flexible Magnetic Sheet and Strip Application Guide*, Akron, OH, 1982.

[29] S. N. Simunovic, "An information approach for parts mating," Ph.D. dissertation, MIT, Cambridge, MA, 1979.

[30] G. C. Burdea, "Two piece jig-saw puzzle robot assembly with vision, position, and force feedback," in *Proc. IEEE-ACM 1987 Fall Joint Computer Conf.* (Austin, TX) pp. 505–511, Oct. 1987.

[31] K. Salisbury, "Active stiffness control of a manipulator in Cartesian coordinates," in *Proc. 19th IEEE Conf. on Decision and Control*, pp. 95–100, 1980.

[32] P. Andreae, "Justified generalization: Acquiring procedures from examples," Ph.D. dissertation and Tech. Rep. AI-TR-834, MIT, Cambridge, MA, Jan. 1985.

[33] R. A. Brooks, "Symbolic error analysis and robot planning," *Int. J. Robotics Res.*, vol. 1, no. 4, pp. 29–68, Winter 1982.

[34] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. J. Robotics Res.*, vol. 5, no. 4, pp. 56–68, 1986.

[35] M. T. Mason, "Mechanics and planning of manipulator pushing operation," *Int. J. Robotics Res.*, vol. 5, no. 3, pp. 53–71, Fall 1986.

[36] M. A. Peshkin and A. C. Sanderson, "The motion of a pushed, sliding object, Part 2: Contact friction," Tech. Rep. CMU-RI-TR-86-7, Carnegie-Mellon Univ., Pittsburgh, PA, 1986.

[37] ——, "The motion of a pushed, sliding object, Part 1: Sliding friction," Tech. Rep. CMU-RI-TR-85-18, Carnegie-Mellon Univ., Pittsburgh, PA, 1985.

[38] ——, "Planning robotic manipulation strategies for sliding objects," in *IEEE Int. Conf. on Robotics and Automation*, pp. 696–701, Apr. 1987.

[39] ——, "Planning robotic manipulation strategies for workpieces that slide," *IEEE J. Robotics Automat.*, vol. 4, no. 5, pp. 524–531, 1988.

[40] J. Hong and H. J. Wolfson, "An improved model-based matching method using footprints," *Proc. Int. Conf. on Pattern Recognition* (Rome, Italy, Nov. 1988).

Grigore C. Burdea (S'87–M'87) has received the engineering degree (valedictorian) in Bucharest, Romania, in 1980, where he subsequently worked as a certified electrical engineer. He received the M.S. and Ph.D. degrees in applied science and robotics from New York University, New York, NY, in 1985 and 1987, respectively.

He joined the Electrical and Computer Engineering Department at Rutgers—The State University of New Jersey, Piscataway, NJ, as an Assistant Professor in January, 1988. His research interests include force feedback control in robotics and particularly dextrous telerobotics.

Dr. Burdea is a member of ASME and The New York Academy of Sciences.

Haim J. Wolfson (M'87) received the B.Sc., M.Sc., and Ph.D. degrees in mathematics from the Tel Aviv University, Tel Aviv, Israel.

From 1985 to 1989 he was an Associate Research Scientist and a Visiting Assistant Professor at the Robotics Research Laboratory of the Courant Institute of Mathematics, NYU, New York, NY. Since 1989 he has been with the Department of Computer Science, School of Mathematical Sciences Tel Aviv University, Israel. His current research interests include computer vision, pattern recognition, artificial intelligence, and robotics.