

## SOLVING JIGSAW PUZZLES BY COMPUTER<sup>\*</sup>

Haim WOLFSON, Edith SCHONBERG, Alan KALVIN and Yehezkel LAMDAN

*New York University, Robotics Research Laboratory, Department of Computer Science,  
Courant Institute of Mathematical Sciences, 251 Mercer Street, New York, NY 10012,  
USA*

### Abstract

An algorithm to assemble large jigsaw puzzles using curve matching and combinatorial optimization techniques is presented. The pieces are photographed one by one and then the assembly algorithm, which uses only the puzzle piece shape information, is applied. The algorithm was experimented successfully in the assembly of 104-piece puzzles with many almost similar pieces. It was also extended to solve an intermixed puzzle assembly problem and has successfully solved a 208-piece puzzle consisting of two intermixed 104-piece puzzles. Previous results solved puzzles with about 10 pieces, which were substantially different in shape.

104-similar  
208-  
combining  
2-104

### Keywords

Computer vision, curve matching, jigsaw puzzle assembly, traveling salesman, assignment, pattern recognition, 2-D shape.

## 1. Introduction

1.1. In this paper, we describe a technique for solving jigsaw puzzles by computer vision and report the experimental results of applying our technique to 104-piece jigsaw puzzles. In our approach, each piece of the puzzle is photographed and digitized and its boundary is calculated. Using only this boundary data, our algorithm calculates a global matching between these puzzle pieces and consequently produces an assembly of the puzzle (as shown in figs. 1 and 5). Similar techniques are applied to a successful solution of an intermixed double puzzle. Two 104-piece puzzles were processed together with no a priori indication that this is the case. The computer program is able to determine that it is dealing with two separate puzzles and then assembles each of them successfully.

puzzle pictures  
problems

method  
details?

<sup>\*</sup>Work on this paper has been supported by Office of Naval Research Grant N00014-82-K-0381, National Science Foundation Grant No. NSF-DCR-83-20085, and by grants from the Digital Equipment Corporation, and the IBM Corporation.

1.2. The jigsaw puzzle assembly problem is regarded as a strenuous test for 2-D curve matching techniques and was studied in [F-G, 6], where a 9-piece puzzle was assembled, using matching techniques based on the Freeman chain code and some heuristics, and in [R-B, 12], where a 4-piece puzzle was assembled using curve matching based on the so-called "boundary-centered polar encoding" technique.

In both these cases, as in our approach, the puzzle is turned upside down, so the only information used is the shape of the pieces (in [F-G, 6] it is called an "apictorial" puzzle). In [R-B, 12], the puzzle pieces were traced on a data tablet. Both [F-G, 6] and [R-B, 12] require a high degree of discrimination between the boundary curves of different puzzle pieces to enable successful puzzle assembly.

1.3. Our approach uses as its basis the Schwartz—Sharir curve matching algorithm (see [S-S, 13]). We use this method to assemble "apictorial" jigsaw puzzles of more than 100 pieces, many of which are similarly shaped (see figs. 3 and 4). We are using commercial children's puzzles (e.g. fig. 1 shows the other side of the "Mickey Mouse & Donald Duck" puzzle by Jaymar), the pieces of which were turned upside down and photographed separately by a black and white camera. This photographic procedure introduces additional noise which does not exist when the input data is obtained artificially.

1.4. Because of these three factors — a relatively large number of puzzle pieces, many strong similarities between boundaries of different pieces, and the additional level of noise — it is difficult to obtain a complete puzzle assembly just from the basic matching algorithm and straightforward heuristics. Hence, the results of the "local" Schwartz—Sharir matching algorithm are taken as input to a second "global" matching which uses these results and assembles the puzzle using further combinatorial optimization techniques. This "global" approach differs substantially from the "local" techniques, which were used in previous works, and enables us to solve puzzles which are by an order of magnitude larger, and which piece shape is much more similar, than in the previous works.

1.5. Since the basic local matching algorithm is described in detail in [S-S, 13], we give only a short overview of it in sect. 4 and this paper is devoted mainly to the description of the global algorithm, which is given in sect. 5, and the double puzzle solution, which is described in sect. 6. However, it should be noticed that the good performance of this local algorithm is an essential component in our puzzle assembly. In fact, from our experiments it is clear that the puzzles with the simpler sizes and shapes used in previous works ([F-G, 6], [R-B, 12]) could be easily assembled using only the local matching algorithm, without any global considerations. Indeed, in a previous experiment, we have managed to assemble a simpler 15-piece puzzle using only local matching.

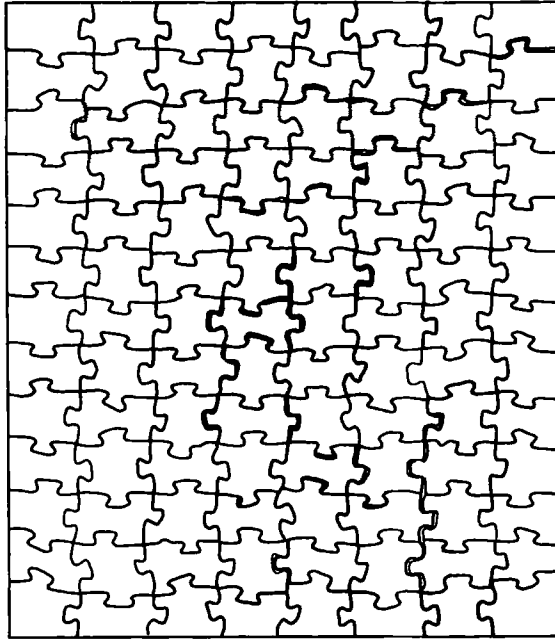


Fig. 1. An assembled 104-piece puzzle.

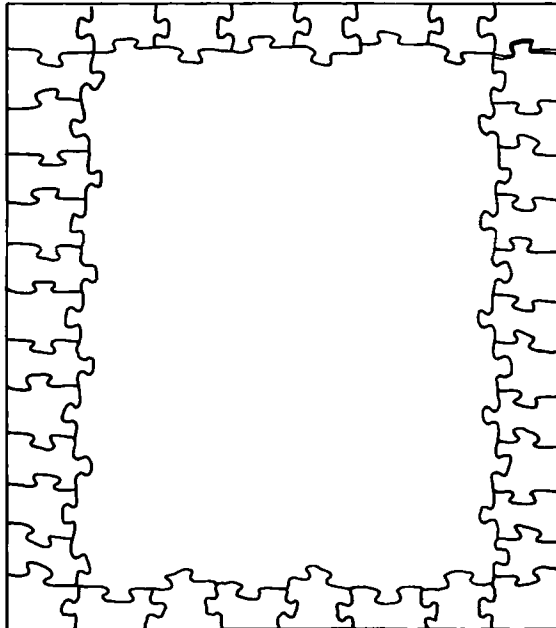


Fig. 2. An assembled frame of the puzzle of fig. 1.

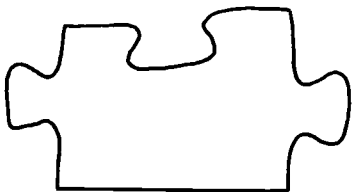
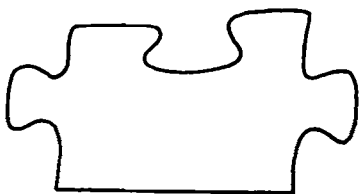


Fig. 3. Two different frame pieces.

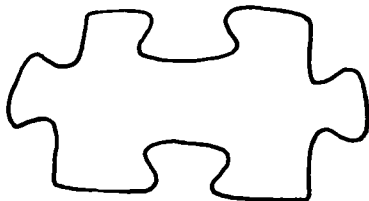
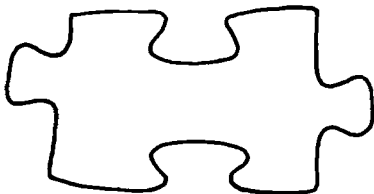
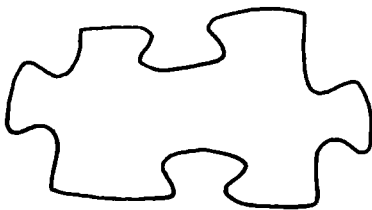


Fig. 4. Three different interior puzzle pieces.

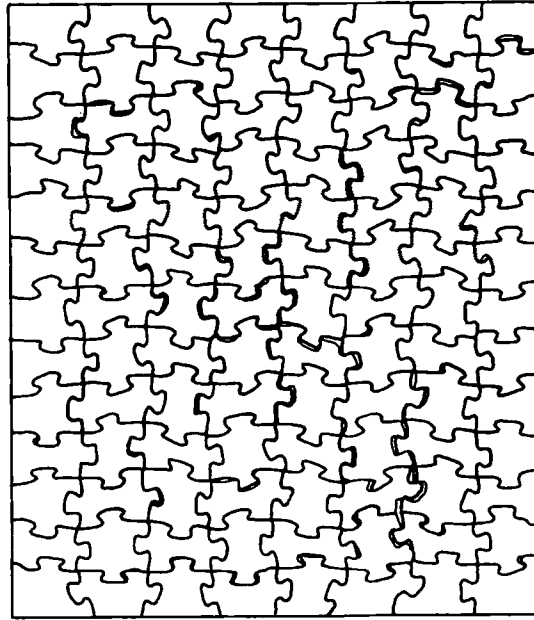


Fig. 5. An assembled 104-piece puzzle.

## 2. Definition of the problem

2.1. We consider two-dimensional rectangular jigsaw puzzles of about 100 pieces (see, for example, fig. 1). Every interior puzzle piece has four neighbours, and every frame piece, except the four corner pieces, has three neighbouring pieces. The puzzle is arranged in a grid. In most of the cases observed, four adjacent pieces meet at a common corner. However, this condition does not hold everywhere (notice at least four "irregular" junctions in fig. 1), and it is not a prerequisite for successful implementation of the proposed algorithm. It will also become clear from the following discussion that the puzzle need not even be rectangular although in such a case we need some other criteria to distinguish the frame pieces of the puzzle.

shape-  
neighbours

2.2. In what follows, we describe the process of the puzzle assembly algorithm step by step, referring specifically to the examples of figs. 1 and 5, which are  $13 \times 8 = 104$  piece jigsaw puzzles of overall size  $18'' \times 13''$ . The intermixed puzzle experiment has used the same two puzzles. We used puzzles of the same size, so there will be no way to distinguish between the pieces of the different puzzles by some "external" parameter, such as size.

### 3. Preprocessing

3.1. The puzzle pieces are photographed one by one by a black and white RCA 2000 camera and the pictures are digitized and thresholded to get a binary image for each puzzle piece.

3.2. The boundary of each piece is extracted from the binary image and a polygonal approximation of it is obtained. This polygonal approximation smoothes the boundary and eliminates some of the noise from it. It is needed in order to apply successfully the [S-S, 13] matching algorithm. All subsequent processing is done on these polygonal approximations of the original boundaries.

3.3. 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. Two different methods have been applied in order to find the four corners of each piece.

A first such method, which has also been used in [K-S-S-S, 8] to find the so-called "breakpoints" in a boundary of a composite scene, first finds all the points on the boundary of the piece at which the tangent has a sharp change in direction and, hence, the second derivative reaches a peak. Since there may be more than four points on the boundary with this property, a heuristic procedure, based on the standard shape of the pieces, is applied to extract the four corners.

The second corner-finding technique exploits the fact that at corners the boundary curve closely approximates a right angle. More specifically, a small portion of a right angle is matched against the boundary curve using the [S-S, 13] matching algorithm, and this gives us a small set of candidates for the four corner points. The distances between these candidate points are measured and certain consistency requirements, such as "almost equal" distance between opposite points, are checked. The four points which pass the "distance consistency" requirements are selected as corner points. Application of this method is in a preliminary stage and some refinements are presently being developed. In the experiments which are described in this paper, we used the first corner-finding technique.

3.4. Pieces having an (almost) straight section between adjacent corners are identified as "perimeter" or "frame" pieces.

### 4. Local matching procedure

The next paragraphs give a short overview of the 2-D curve matching algorithm of [S-S]. For a more detailed exposition, see [S-S, 13].

4.1. Take two curves  $C$  and  $C'$  in the plane and assume that  $C$  is a translated and rotated subcurve of  $C'$ . Both  $C$  and  $C'$  are assumed to have been smoothed (i.e. they are polygonal approximations of the original curves) and parametrized by arc length  $s$ .

The matching we seek calls for determination of the offset  $s_0$  and the Euclidean transformation  $E$  for which the curves  $EC(s)$  and  $C'(s + s_0)$  are closest to one another in the  $L^2$  norm. To be more specific, we represent each of the curves  $C, C'$  by a sequence of evenly spaced points on it, and let these sequences be  $(u_j)_{j=1}^n$  and  $(v_j)_{j=1}^m$ , respectively. Assume first that both curves have the same starting point (i.e.  $s_0 = 0$  and, hence,  $m \geq n$ ). Matching thus amounts to finding a Euclidean motion  $E$  of the plane which will minimize the  $\ell^2$  distance between the sequences  $(Eu_j)_{j=1}^n$  and  $(v_j)_{j=1}^m$ :

$$\Delta = \min_E \sum_{j=1}^n |Eu_j - v_j|^2.$$

To simplify the calculation, first translate  $C$  so that

$$\sum_{j=1}^n u_j = 0.$$

Next, write  $E$  as  $Eu = R_\theta u + a$ ,  $R_\theta$  denoting a counterclockwise rotation by  $\theta$ . In such case, as it is shown in [S-S, 13], the best match is obtained when

$$a = \frac{1}{n} \sum_{j=1}^n v_j$$

and  $\theta$  is the negation of the polar angle of  $\sum u_j \bar{v}_j$ , where the vectors  $u_j, v_j$  are regarded as complex numbers  $u_j, v_j$ . The least-square distance for this best match is given by

$$\Delta = \sum_{j=1}^n |v_j|^2 - \frac{1}{n} \left| \sum_{j=1}^n v_j \right|^2 + \sum_{j=1}^n |u_j|^2 - 2 \left| \sum_{j=1}^n u_j \bar{v}_j \right|. \quad (*)$$

If the curves do not have the same starting point, we have to match the sequence  $(u_j)_{j=1}^n$  to each of the contiguous subsequences  $(v_{j+d})_{j=1}^n$  of the sequence  $(v_j)_{j=1}^m$ , for  $d = 0, \dots, m-n$ . For each such  $d$ , (\*) thus becomes

$$\Delta(d) = \sum_{j=d+1}^{d+n} |v_j|^2 - \frac{1}{n} \left| \sum_{j=d+1}^{d+n} v_j \right|^2 + \sum_{j=1}^n |u_j|^2 - 2 \left| \sum_{j=1}^n u_j \bar{v}_{j+d} \right|.$$

We seek the minimum of the values  $\Delta(d)$ ,  $d = 0, \dots, m-n$ , which can be found in

time  $O(m \log m)$ , using the fast Fourier transform algorithm for computing the convolutions

$$\sum_{j=1}^n u_j \bar{v}_{j+d}.$$

*Remark:* Although in most junction points, the matching curves have the same starting point, we *deliberately* ignore this information in order to make the performance of the algorithm more general.

4.2. The local matching procedure described above is applied to every pair of "quarter boundary" subcurves of the puzzle pieces (see 3.3). Hence, if we have an  $N$ -piece puzzle, we get a  $4N \times 4N$  symmetric matrix of the best matching scores between every two such sides.

It will be clear from the description of the global algorithm (see sect. 5) that we do not have to compute all the entries of this matrix, since the frame pieces are treated separately; nevertheless, we still require  $O(N^2)$  matchings.

## 5. The puzzle assembly algorithm

5.1. The 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 the assembly of the entire puzzle.

### FRAME ASSEMBLY

5.2. Assembly of the puzzle frame may be viewed as the assembly of a one-dimensional puzzle.

Once the frame pieces have been recognized (see 3.4), their straight line side is known. Suppose that we name the sides of the piece in the order of their appearance counterclockwise — bottom, right, top and left sides — where the straight line is the bottom side (see fig. 3). For the four corner pieces, we will 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 and a left side.

5.3. Define a matrix  $M(i, j)$ ,  $i, j = 1, \dots, K$ , where  $K$  is the number of the frame pieces, 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  $j$ . 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  $i$ th row.

Our task is to find  $K$  entries in  $M$ , one in each row and one in each column, in such a way that the sum of these  $K$  entries is minimal and the  $K$ -permutation  $P$



is a cycle. Suppose that these entries have been determined to be  $j_1, \dots, j_K$ , where  $j_i$  is the entry in the  $i$ th 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, [C, 4] or [L-L-RK-S, 10]), and is usually formulated as follows:

*Given  $K$  cities and the distances between each pair of them, 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  $i$ .\*

5.4. The *traveling salesman* problem is known to be NP-complete (see [G-J, 7]), but for small  $K$ 's (less than 100), there are efficient algorithms that solve it (see, for example, [L-L-RK-S, 10]). (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, [R, 11], [C, 4]);
- (b) algorithms which always find the optimal solution, but in some cases may be very time consuming (see [B-M, 1]).

We actually use the algorithm of type (a), which was published in [F, 5]. This type of algorithm is preferable for our purpose, because it guarantees fast performance. Since we plan to implement our solution by a robot (see [B-W, 3]), possible errors in the proposed solution will be corrected by the following interactive procedure:

- (1) Apply the *traveling salesman* algorithm to the problem defined by matrix  $M$  and feed the solution to the robot.
- (2) Check the proposed solution by the robot and return to the computer information about correct, incorrect and (possibly) undecidable matches between suggested neighbouring pieces.
- (3) If all the matches are correct, go to the next step. If some of the matches are incorrect, change matrix  $M$  according to the information obtained from (2), namely, if a match  $(i, j)$  was correct, assign zero to the  $(i, j)$  entry of the matrix and assign "almost infinity" to all other entries of row  $i$  and column  $j$ ; if a match  $(i, j)$  was incorrect, assign "almost infinity" to the entry  $(i, j)$ , and if a match  $(i, j)$  is undecidable, make no change in the matrix. Then we go back to (1).

\**Added in proof:* It seems to be advantageous to apply the so-called "bottleneck" *traveling salesman* algorithm to our problem, since we are interested in minimizing the maximal mismatch between the pairs of frame pieces. We thank one of the referees for this observation.

This iterative procedure has the advantage of quickly reducing the "practical dimension" of the *traveling salesman* problem being considered, especially if a large percentage of the suggested matches was correct. We believe that a very small number of such iterations will be needed in order to obtain the correct solution. As was mentioned above, the iterative procedure will be implemented using a robot. At this stage, our experiments were based purely on computer vision equipment, so no feedback was available, and we have obtained the correct frame assembly after the first application of the *traveling salesman* algorithm on the original matrix. This happens with high probability, because the matrix  $M$  is not arbitrary, but strongly favourable to its optimal solution by virtue of the good performance of the local matching algorithm. In most cases, we therefore expect to obtain the optimal solution by using the algorithm of [F, 5] without any feedback. As mentioned, this has been the case in our experiments.

5.5. If we take the *traveling salesman* problem and drop the condition that  $P$  should be a cycle, we obtain another well-known problem, which is called the *assignment* or *bipartite matching* problem (see [C, 4], [L, 9]). There is an efficient solution to this problem with a polynomial (i.e.  $O(K^3)$ ) time complexity. Many type (b) solutions to the *traveling salesman* problem are based on this algorithm (see [L-L-RK-S, 10]), and in our case, we believe that in most applications even this less constraining *assignment* algorithm will find the correct assembly of the outer frame.

In sect. 6, we describe the use of the *assignment* algorithm in order to assemble the frames of a number of intermixed different puzzles simultaneously.

*Remark:* As is seen from the above description of the algorithm, the rectangular form of the puzzle is *inessential* to correct arrangement of the frame, since we deliberately ignore the fact that the right and left sides of matching pieces begin from the same bottom starting point, and the bottom straight line is used only to recognize the frame pieces and their outer side. All that we actually need in the case of non-rectangular puzzles is an alternative method of obtaining the same information. For example, the puzzle could be circular or the outer frame side could be simply marked as such in some manner.

## ARRANGEMENT OF THE PUZZLE INTERIOR

5.6. After complete arrangement of the outer frame (see fig. 2), we go on to the arrangement of the interior pieces. Basically, this is done by considering the four interior corners of the frame (see fig. 2) 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.

5.7. 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.

5.8. However, in our situation, in which the boundaries of the pieces are quite similar, even when two sides are being matched (see fig. 4), a greedy algorithm is unlikely to succeed. Hence, some backtracking or branch and bound algorithm is necessary. For this we use the following approach:

(1) Corners are processed sequentially, beginning with the lower left side of the puzzle interior and advancing within each row to the right. (The 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 non-essential.)

(2) For the first corner, all the puzzle pieces which are not frame pieces are located at this corner in all possible rotations, 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.

(3) At the second corner, the same procedure is repeated for every one of the KBEST partial solutions which passed the previous stage, and only KBEST overall best solutions are passed to the next stage.

(4) 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.

Of course, the number of 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 experiments (figs. 1 and 5), we used KBEST = 200, which was kept the same at every stage. Actually, the correct solution in the puzzle of fig. 1, for example, always lay among the ten best solutions.

5.9. The above described algorithm is polynomial, but it does not assure the correct solution in all the cases. The complexity of the proposed puzzle assembly algorithm is computed as follows. Given an  $N$  piece puzzle with  $K$  frame pieces and  $O(m)$  sample points on a boundary of a piece (see paragraph 4), the frame assembly using the *assignment* algorithm requires  $O(\max(K^3, K^2 m \log m))$  operations, and the puzzle interior assembly requires  $O(N^2 \times \text{KBEST} \times m \log m)$  operations. If we apply the *traveling salesman* algorithm to assemble the frame, the running time depends on

the specific heuristic which is used (see discussion in 5.4). However, as was mentioned before, the distance (score) matrix in our case is not arbitrary, but is assumed to be a priori biased in favour of the correct solution due to the performance of the local curve matching algorithm. Hence, fast convergence to the correct result is expected.

5.10 In the next few paragraphs, we describe the way in which simultaneous matching of two sides of a piece is accomplished.

The most accurate way is to arrange the frame (or a given partial solution), take a picture of the relevant corner, and then try to match this curve with the boundary curves of the remaining pieces. However, this method is too tedious for a computerized assembly.

A reasonable approximation to such a method is to use available information about the relative matching angles and displacements of the pieces and to calculate the curve formed by joining two sides at a corner. This method has the disadvantage of accumulating angular errors, and may result in a slightly distorted link-up between the curves meeting at a corner junction.

To avoid this accumulation of errors while not losing the information that the two curves being matched at a corner must belong to the same piece, we compute the sum of the scores of both sides when they are taken independently, and we add to this sum a penalty score which is proportional to the excessive shift of the side curves, if they do not meet. This procedure enables us to precompute the matrix of all possible best matches and their relative displacements for every pair of curves (see 4.2), eliminating the need for a local matching procedure to be re-invoked within the main loop of the global algorithm.

The puzzle in fig. 1 was solved correctly even without using this penalty score on wrong displacements, i.e. by considering both sides independently; however, to solve correctly the puzzle in fig. 5, we had to use the penalty score. We assume that the difference lies in the accuracy of the data acquisition step.

## 6. Solution of an intermixed double puzzle

6.1. To test our methods in a still more strenuous manner, two 104-piece puzzles (figs. 1 and 5) were intermixed and treated as a single 208-piece puzzle.

Preprocessing and local matching were done in the same way as described in sects. 3 and 4.

However, the puzzle assembly algorithm was changed to reflect the new fact characterizing this sort of intermixed puzzle problem, since this time we had to drop the (wrong) assumption that one connected puzzle has to be assembled, i.e. that the frame permutation should consist of a single cycle.

6.2. The frame pieces were recognized and separated as before, but the frame assembly was not done using the *traveling salesman* algorithm, which always produces one cycle of a permutation  $P$  (see 5.3 and 5.4), but by the *assignment* algorithm

(see 5.5). The number of the cycles in the permutation obtained should then be equal to the number of different puzzles being processed simultaneously, and the size of the cycles correspond to the size of the puzzle frames.

Once the frames of the different puzzles are obtained, we can arrange their interior by the methods described in 5.6–5.8. Note, however, that since we have no way to distinguish between the pieces of the different puzzles (except the frames), the assemblies of the different interiors must proceed in parallel.

6.3. In our experiment, we obtained two (correct) frame cycles using the *assignment* algorithm and then arranged the interior of one of them by the methods of 5.6–5.8, when all the remaining puzzle pieces (of both puzzles) were considered as candidates for the vacant places. Once one of the puzzles was assembled correctly, we were left with the problem of assembling only one 104-piece jigsaw puzzle, as done before. This approach was chosen merely because it required only minimal changes in the existing single-puzzle assembly program; of course, more sophisticated technical details could be developed.

6.4. It is obvious that our approach is completely general and may be used for the assembly of any number of puzzles. The number of different cycles of the permutation  $P$  (see 5.3) which is obtained by the *assignment* algorithm (see 5.5) determines the number of different puzzles to be assembled and their frames. Now, the interiors of all the puzzles have to be arranged simultaneously, when all the interior pieces "compete" on all the corners in the different puzzles.

6.5. The complexity of this algorithm is again

$$O(\max(K^3, K^2 m \log m) + O(N^2 \times \text{KBEST} \times m \log m),$$

where  $N$  is the total number of puzzle pieces,  $K$  is the number of frame pieces, and the number of sample points on a boundary curve of a piece is  $O(m)$  (see 5.9).

*Remark:* In the experiment described above, we used a local matching score which takes into account the relative displacement of the puzzle pieces, both for the solution of the interior and the frame (see 5.10).

## 7. Future experiments and research

7.1. It is quite obvious that the same methods can be applied to larger puzzles.

7.2. For puzzles which do not have a grid-like form, and for which the boundary curve can not be divided into four subcurves, other heuristics need to be developed. This more general case suggests the following fundamental:

*Problem:* given two curves, find the longest matching subcurve which appears in both curves.\*

\**Added in proof:* This problem was recently solved in [W, 14].

A good algorithm for solving this problem would make it possible to apply our global algorithm to arbitrary puzzles, without additional heuristics.

## Acknowledgements

The authors thank Micha Sharir and Marc Bastuscheck for helpful discussions.

## References

- [1] [B-M] M. Belmore and J.C. Malone, Pathology of traveling-salesman subtour-elimination algorithms, *Oper. Res.* 19(1971)278.
- [2] [B-N] M. Belmore and G.L. Nemhauser, The traveling-salesman problem: A survey, *Oper. Res.* 16(1968)538.
- [3] [B-W] G. Burdea and H. Wolfson, Automated assembly of a jigsaw puzzle using the IBM 7565 Robot, Tech. Rep. No. 188, Comp. Sci. Div., Courant Inst. of Math., NYU (1985).
- [4] [C] N. Christofides, *Graph Theory* (Academic Press, 1975).
- [5] [F] Z. Fencel, Routing problem, *CACM Algorithm* 456.
- [6] [F-G] H. Freeman and L. Garder, Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition, *IEEE Trans. on Electronic Comp.* EC-13, 2(1964)118.
- [7] [G-J] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman and Co., 1979).
- [8] [K-S-S-S] A. Kalvin, E. Schonberg, J.T. Schwartz and M. Sharir, Two dimensional model based boundary matching using footprints, Tech. Rep. No. 162, Comp. Sci. Div., Courant Inst. of Math., NYU (1985).
- [9] [L] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids* (Holt, Rinehart and Winston, 1976).
- [10] [L-L-RK-S] 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* (Wiley, 1985).
- [11] [R] T.C. Raymond, Heuristic algorithm for the traveling-salesman problem, *IBM J. Res. Develop.* 13, 4(1969)400.
- [12] [R-B] G.M. Radack and N.I. Badler, Jigsaw puzzle matching using a boundary-centered polar encoding, *Computer Graphics and Image Processing* 19(1982)1.
- [13] [S-S] J.T. Schwartz and M. Sharir, Identification of partially obscured objects in two dimensions by matching of noisy "characteristic curves", Tech. Rep. No. 165, Comp. Sci. Div., Courant Inst. of Math., NYU (1985).
- [14] [W] H. Wolfson, On curve matching, Tech. Rep. No. 256, Comp. Sci. Div., Courant Inst. of Math., NYU (1986).