Pattern Recognition Letters 29 (2008) 1924–1933

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Solving jigsaw puzzles using image features

Ture R. Nielsen, Peter Drewsen, Klaus Hansen *

Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 Copenhagen, Denmark

ARTICLE INFO

Article history:
Received 10 December 2006
Received in revised form 28 May 2008
Available online 14 June 2008

Communicated by R. Davies

Keywords:
Jigsaw puzzle solver
Edge matching
Piece classification
Border similarity measure
Co-occurrence matrix

ABSTRACT

In this article, we describe a method for automatic solving of the jigsaw puzzle problem based on using image features instead of the shape of the pieces. The image features are used for obtaining an accurate measure for edge similarity to be used in a new edge matching algorithm. The algorithm is used in a general puzzle solving method which is based on a greedy algorithm previously proved successful. We have been able to solve computer generated puzzles of 320 pieces as well as a real puzzle of 54 pieces by exclusively using image information.

Additionally, we investigate a new scalable algorithm which exploits the divide and conquer paradigm to reduce the combinatorially complex problem by classifying the puzzle pieces and comparing pieces drawn from the same group. The paper includes a brief preliminary investigation of some image features used in the classification.

1. Introduction

Solving puzzles has been a popular pastime for ages and several researchers have recently looked into solving puzzles automatically by a computer. However, the problem is hard to solve for machines as compared to the relative ease with which humans can solve even very large puzzles. The computer typically does not have the same certainty of the correctness when matching a pair of puzzle pieces, but rather a likelihood for the correctness.

While the puzzle problem is a justified field of research in itself, research on puzzles might also be justified by the potential for real life applications of some of its components. The automatic solving of jigsaw puzzles deals with problems such as boundary detection, shape matching and texture comparison, which are of general interest in the fields of computer vision and pattern recognition. This paper is based on the assumptions that pieces have rigid edge curves constrained to a plane, and that the puzzle has no missing parts and has the topology of a plane. These assumptions are, however, not crucial for all sub-problems and the edge matching can for instance be extended for use in piecing together fragments of pictures or objects such as archaeological artifacts.

Previous works have all focused on edge matching using shape rather than image features. We therefore feel that using image features in solving jigsaw puzzles has not really been investigated and have chosen to focus on this aspect in our work. Using both shape and image information is likely to provide the best matching results, as image features may improve the precision of the edge matching measure and may be used in an initial grouping of the pieces to reduce the complexity when comparing pieces. We have, however, chosen to exclude entirely the shape aspect to show the strength of using image features and to prove that it is possible to solve jigsaw puzzles without shape information. This also generalizes the puzzle problem to include puzzles where an edge description is either not available or non-unique. In our specific case it allows us to solve jigsaw puzzles which only employ a small set of different shapes, as well as puzzles with rectangular pieces.

It should be noted that our edge matching can be expanded to include shape information, and we expect this will improve the performance on typical puzzles.

This article is based on our previous work (Nielsen et al., 2006) (in Danish) on the same subject. An important addition is the testing of our algorithm on real puzzle pieces.

1.1. Problem definition

The puzzle problem is the problem of assembling a jigsaw puzzle so that all pieces fit together forming a picture. We assume that the pieces each have four sides and are arranged in a rectangular grid. Each side of a piece can either be concave, convex or have a straight edge. Border pieces have one or two straight edges. Unlike previous definitions, however, we do not require non-straight sides to be unique.

To demonstrate the power of being independent of the uniqueness of the shape, we also expand our definition to encompass puzzles with perfectly rectangular pieces. Such puzzles have the additional property that they are easy to generate when we are

* Corresponding author. Tel.: +45 35 32 14 00; fax: +45 35 32 14 01.
E-mail address: khan@diku.dk (K. Hansen).

0167-8655/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2008.05.027

testing our methods. However, rectangular pieces have no distinctive edge, which makes it practically impossible to identify border pieces. For this reason, we instead require that border pieces be identifiable from image features, for instance by having an outline or frame in the image, or that a person identify them. The final tests of our algorithm will, however, be performed on classical pieces of both real and computer generated puzzles.

1.2. Related work

The first article concerning the puzzle problem is Freeman and Gardner (1964), who worked with puzzles without pictures. Despite not actually being implemented, their work laid the foundation for many subsequent papers. Their method attempts to identify critical points along the edge that can be used for segmentation. In this way a measure is calculated for how well the pieces fit together, also known as *partial boundary curve matching*. Radack and Badler (1982) also employed partial boundary curve matching, this time using polar coordinates. Wolfson et al. (1988) developed an algorithm for solving puzzles using two-dimensional *Schwartz–Sharir curve matching* and optimized search trees, and managed to solve puzzles of up to 104 scanned pieces.

Kosiba et al. (1994) proposed the first method using both the image on and the shape of the pieces, thus managing to correctly solve small puzzles as well as puzzles of up to 54 pieces with "satisfactory" results. The idea of using both image and shape was further employed by Chung et al. (1998), who also managed to solve puzzles of 54 pieces, and later by Yao and Shao (2003), who managed to solve puzzles with "dozens" of pieces.

To the best of our knowledge the largest puzzle solved to date by a computer has 200 pieces, and was solved using the method introduced by Goldberg et al. (2004). Their method employs only the shape of the pieces, and instead of a search tree they use a greedy approach, building the puzzle from the corners of the solved border. This heuristic solves the puzzle problem in polynomial time, thus greatly reducing the complexity.

1.3. Solution outline

The automatic assembly of a puzzle can be divided into five separate problems:

- preprocessing,
- edge matching,
- solving the border,
- solving the interior,
- laying the pieces and presenting the solution.

Preprocessing generally consists of scanning real puzzles, separating the individual pieces and then rotating them to achieve the proper alignment. There are several challenges to this process for real puzzles, which we discuss in Section 4.

Edge matching is performed by matching an edge of one piece to an edge of another piece, thereby obtaining the likelihood of the two pieces fitting together at those edges. This is described in Section 2, where we propose a new method solely using image features. This local match can then be used in assembling the border as well as the interior pieces.

Having devised a method of matching pieces, the *solution* to the puzzle problem lies in the ordering of the pieces to obtain the highest global match. This is a combinatorial problem whose complexity is $O(n!)$, when searching through the entire solution space. Due to the high complexity, the problem is often split into the separate problems of solving the border and solving the interior. Because the complexity of the interior puzzle remains the same, a heuristic is used to reduce the problem. In Section 3 we will briefly discuss a heuristic for solving the border as discovered by Wolfson et al. (1988), as well as a highly successful heuristic for solving the interior by Goldberg et al. (2004).

Finally, the puzzle solver presents the solution by *displaying the assembled pieces*, and this includes minimizing overlaps and holes through corrections to the orientation of the pieces. While some algorithms perform this step during the solution stage to further evaluate edge similarity, it can be considered a separate sub-problem and be performed after the topological solution is known. As this problem is not the focus of our work, we will use a simple algorithm, which only rotates pieces in increments of 90° (given by the topological solution) and does not try to minimize the error through corrections to the orientation.

1.4. A brief remark on the test data

We use two sets of puzzles for testing. Two images are used to create synthetic puzzles for testing edge matches. One image (`landscape`) has groups of relatively homogeneous pieces and the other (`construction`) has varying colors and textures. The images are shown in Fig. 1.

During our edge matching tests we divide the images into puzzles of rectangular and classical non-rectangular pieces ranging in numbers from 56 to 504. The pattern used for cutting these pieces is shown in Fig. 2, and is repeated vertically and horizontally, thus creating at most 16 unique shapes for interior pieces and at most 32 unique edges.

In addition, two real puzzles of 24 and 54 pieces have been scanned (see Section 4 for details). Fig. 10 shows part of the solution found by our algorithm for the 54-piece puzzle (`Benjamin`). The complete set of test puzzles may be found at the DIKU Image Group's FTP server (2008).

2. Edge matching

How well two puzzle pieces match along a common edge is expressed as a *similarity measure*. As mentioned above, there have only been a few experiments on using image features for matching pieces and none which rely only on this. Kosiba et al. (1994) and Chung et al. (1998) both sampled small areas at intervals along the edge using local image features such as mean, variance and histogram difference to calculate a similarity measure. Yao and Shao (2003) used the integration degree[1] on subdivided strips all along the edge for their solution.

Common for these methods is that shape matching is used as well. In this article, we propose a method that solely uses image features. The method is different from previous methods in that it only considers a single-pixel wide continuous strip for each edge as described in Sections 2.1 and 2.2. It is closely related to directional edge detection in that two pieces are considered a likely match if there is little or no gradient at the common edge between them.

The sampling method and the similarity measure are interconnected, and we first describe the similarity measure (Section 2.1), assuming we have derived straight image strips from each edge. In Section 2.2 we then define a way of sampling the edge of the pieces to generate these strips.

2.1. Similarity measure

Imagine two vertical image strips representing the edges of two pieces that are to be compared. By placing them side by side and

[1] The integration degree is a form of variance test, and is defined by Yao and Shao (2003) from the separability of image features, which in turn was described by Otsu (1979).

(a) landscape (b) construction

Fig. 1. Images used for puzzle generation (available at the DIKU Image Group's FTP server (2008)).

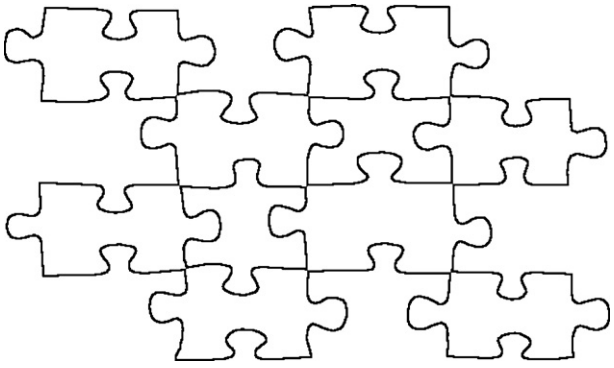


Fig. 2. Template used for cutting images.

running a vertical edge detector, one should expect to get a significant *vertical* edge at the boundary between the pieces if the two pieces do not fit together. Conversely, the edge detector only produces noise (created by the small variance in the texture itself) if the two pieces do fit together. This is demonstrated in Fig. 3 using the Sobel operator, defined in Sonka et al. (1999) among others, for vertical edge detection.

While edge detection is usually performed on gray scale images or the intensity component of color images, it is possible to detect edges in any given image feature f, such as hue, saturation, intensity or red, green, blue, etc. by computing the gradient between neighboring values.

For the proposed method, we are only interested in edges at the common boundary, and we therefore use the following filter-based on the Sobel operator to obtain a response vector, $\boldsymbol{V}_f$, for each feature, which in this case is one of the color channels.

$$H = \begin{vmatrix} -1 & 1 \\ -2 & 2 \\ -1 & 1 \end{vmatrix}.$$

Given the two edge strips (1 pixel wide) as column vectors, $\boldsymbol{s}_f^{\text{left}}$ and $\boldsymbol{s}_f^{\text{right}}$ for feature f, the response vector is calculated by convolving the filter with a horizontal concatenation of the strips.

$$\boldsymbol{V}_f = H \circ (\boldsymbol{s}_f^{\text{left}} | \boldsymbol{s}_f^{\text{right}}).$$

Let H_{ij} be the element of the filter in the ith row and jth column and $\boldsymbol{s}_{f,i}$ the ith element of the border strip for feature f. The response vector $\boldsymbol{V}_f = (v_{f1}, v_{f2}, \ldots, v_{fn})^{\mathrm{T}}$, where n is the size of the strips, can then also be written as:

$$v_{fi} = \sum_{j=1}^{3} H_{j1} \cdot \boldsymbol{s}_{f,i+j-2}^{\text{left}} + H_{j2} \cdot \boldsymbol{s}_{f,i+j-2}^{\text{right}}, \quad i = 1, \ldots, n.$$

In case the two strips are not perfectly aligned, it is possible to use a larger filter. Alternatively the strips may be shifted relative to each other to find a local minimum response.

From the response vector we calculate the mean square:

$$\overline{\boldsymbol{V}_f^2} = \frac{1}{n} \sum_{i=1}^{n} v_{fi}^2.$$

To get a similarity measure in the range $[0, 1]$, we use the following equation to obtain a value which will give an rapid fall-off:

$$\text{Sim}(\boldsymbol{s}_{\text{left}}, \boldsymbol{s}_{\text{right}}) = \frac{1}{1 + \alpha \sqrt{\sum_{f=1}^{\xi} \overline{\boldsymbol{V}_f^2}}}, \tag{1}$$

where ξ is the number of features (color channels). The α parameter is a scaling factor that controls how rapidly the curve drops off. As the ordering of edge matches is not affected by the choice of α, its value is only of interest in future thresholding schemes as described in Section 5.

In order to find the image features best suited for the similarity measure, we have tested combinations of HSI (hue, saturation, and intensity) and RGB (red, green, and blue). For a given edge, if the true match is also the edge with the highest similarity, it is considered a *true positive* (TP). Otherwise it is considered a *false negative* (FN). As indicator of reliability we calculate the *true positive rate* (TPR) for all possible edge pairs defined as follows (note that TP + FN is the total number of interior edges):

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2}$$

Table 1 shows the results of this test. Here, the edges have been sampled at a depth of 3 pixels for pieces approximately 150×150 pixels in size, which was found to be the optimal depth for pieces of this size.

It can be seen that a combination of all the HSI color components yields the highest true positive rate for all puzzle pieces, and these are therefore considered the most reliable features to use.

2.2. Edge sampling

When using classical puzzle pieces with either concave or convex edges we need a method to extract an edge strip $\boldsymbol{s}$ that we can use as parameter in the comparison method described above.

When dealing with possibly imperfect representations of puzzle pieces there are several factors to consider. One important problem is that scanned pieces often have leftover cardboard fragments from the cutting process, which can cause a digital segmentation to leave artifacts along the edges of pieces. In addition, imprecise segmentation can also cause other problems near piece edges. Finally, some images of pieces may even include the side of the pieces because the cut has an oblique angle with the camera axis. See Section 4 for more about some of the problems encountered when using real pieces.

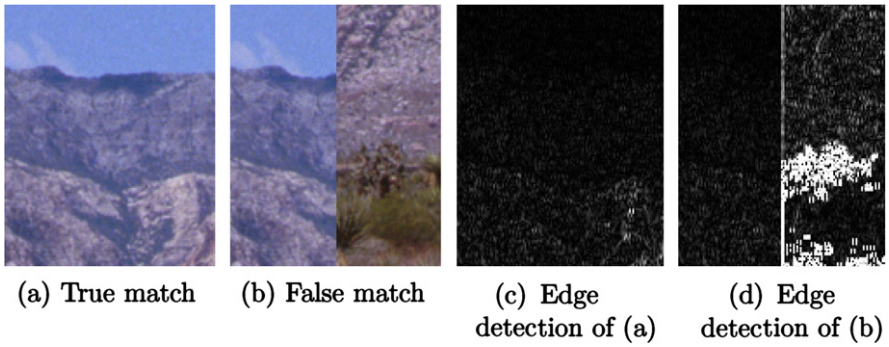


Fig. 3. (a) and (b) show rectangular pieces at their joining edge. In (a) the pieces are correctly matched, whereas in (b) the match is incorrect. (c) and (d) show the pieces when a vertical edge detector has been applied to the hue color component.

Table 1
True positive rates for all edges matched against all other edges in three puzzles (see Figs. 1 and 10)

Features	Puzzle (# pieces)		
	Construction (100)	Landscape (100)	Benjamin (54)
H	85%	97%	68%
S	78%	96%	62%
I	86%	97%	25%
H + S	93%	100%	77%
H + S + I	98%	100%	82%
R + G + B	44%	46%	75%

Best results are obtained when combining hue, saturation, and intensity color components. The low score for the combination of I and Benjamin is possibly because of the automatic brightness control in the camera.

We note that simply following the edge of a piece pixel by pixel, choosing the next pixel from the 8-neighborhood, will result in an uneven sampling on each edge of two matching pieces. This means that pixels on one edge may drift relative to the adjacent pixels on the other edge due to the different number of samples caused by the curvature of the edge. This problem is demonstrated in Fig. 4. To avoid the problem of uneven sampling we have used a method, which should generate the same number of samples for matching edges.

Our method first determines the location of the "corners" of the piece as demonstrated in Fig. 5. We limit the search to a sub-image, the location of which is in the range from 0% to an ample 25% of the length of the sides of the enclosing image in both directions. The sub-image is shifted 15% away from the image boundary if the corresponding edge is convex. This shift value is based on the protrusion sizes in the puzzles we used.

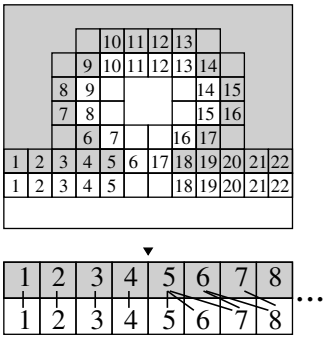


Fig. 4. Example of uneven sampling of two matching edges (white and gray), by sampling the pixels along the edges so that each sample is in the previous sample's 8-neighborhood. The two generated strips are unsuited for a pixel by pixel comparison, as indicated by the thick lines connecting adjacent border pixels.

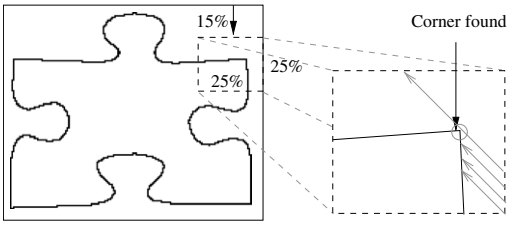


Fig. 5. The corners of a piece are found by progressive diagonal scan lines in a sub-image of the piece.

The corner points are used as start and end points of the edge in question. We proceed by computing points along the edge so that they are placed exactly on the contour of each piece on the outer edge of every pixel. This is shown in Fig. 6. The method ensures the same number of samples for perfectly matching concave and convex edges.

From each of the points along the contour we estimate a tangent using the secant spanned by points on either side as is also seen in Fig. 6. The distance between the points used for the estimates should depend on the size of the puzzle piece, as greater distance between points will make the tangent less sensitive to smaller bumps along the edge. In our own experiments we calculated the tangent from the secant spanning seven points for pieces roughly 100×150 pixels in size.

After the tangent is estimated our method calculates sample points perpendicular to the tangent starting at some initial depth. From these points the value for the edge strip is calculated by linear interpolation using the neighboring pixels. This principle is illustrated in Fig. 7 and is shown in Fig. 8 for a real piece, using a variable depth.

When using this method, non-matching edges are likely to generate edge strips of different length, since both the corners and the curvature of the edge vary from piece to piece. Furthermore, even matching edges might generate strips of slightly different length due to imperfections in the segmentation of the pieces. Aligning the strips for edge matching is a special case of the string matching or sequence alignment problems for which methods like dynamic time warping and dynamic programming provide solutions.

We have, however, chosen a simpler approach, because we expect the texture details to cover several pixels,[2] the imperfections in the edges due to the production and scanning processes to be small, and the corners determining the beginning and end of each strip to be well defined. The edge matching is furthermore designed to distribute errors evenly over a relatively short edge strip.

[2] The smoothing described in Section 4 contributes to this.

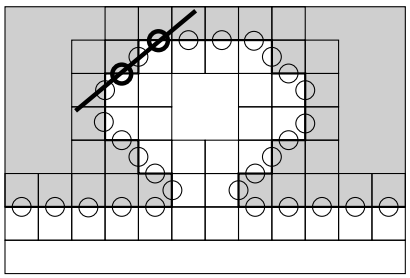


Fig. 6. Example: The circles denote points on the common contour between matching pieces (white and gray). The points are used for computing a secant, which is used as an approximated tangent for the border curvature.

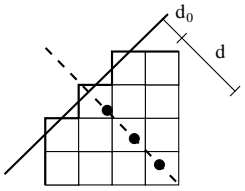


Fig. 7. The edge is sampled perpendicularly to the tangent starting at an initial depth d_0 and to a total depth of $d_0 + d$. Edge strip values are calculated through linear interpolation.

The method we have used to solve the problem of different length is thus simply centering the two strips relative to each other and truncating the longer strip by an equal amount in each end. Experiments with more advanced methods have not yielded any significant improvement in the results.

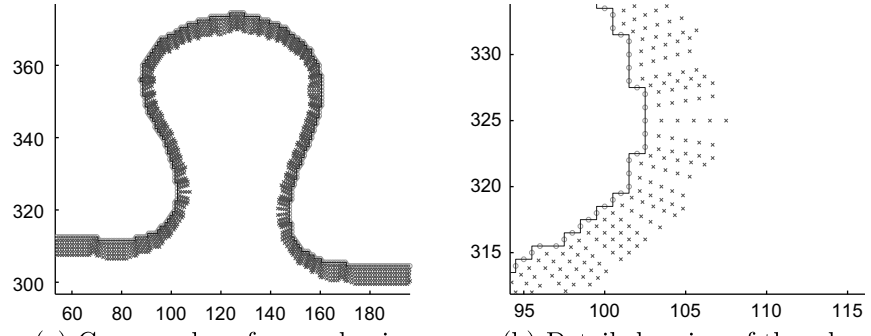
2.3. Edge matching results

Edge matching is done by computing a similarity measure that is strongly indicative of how well two pieces fit together. For one edge compared against a number of others, this value should be close to 1 for the matching edges and rapidly decreasing to zero for the remaining pieces.

Fig. 9 contains two graphs from our tests. They show the similarity measures for three selected border strips matched against all other border strips. The edges have been sorted according to the similarity measure to show that the similarity drops quickly after the best match, and divides edge matches into three distinct groups making thresholding easier.

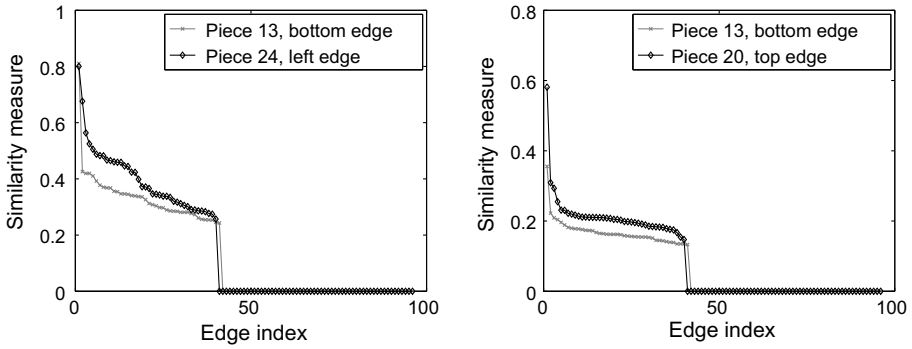
As the similarity measure is crucial to the success of the puzzle solver and we have furthermore compared our method to other methods, such as mean and variance of hue, saturation, and intensity features as proposed by Kosiba et al. (1994), median and histogram difference of hue and saturation as proposed by Chung et al. (1998), and finally integration degree of red, green, and blue color components as performed by Yao and Shao (2003). In addition, the integration degree has also been used on the hue, saturation, and intensity components.

The feature comparison in these methods have been implemented to the best of our abilities as described by the authors, whereas the sampling windows have been created using our own generated edge strips (without applying a Gaussian filter) at an optimal depth, width, and number of subdivisions for each



(a) Convex edge of a puzzle piece. (b) Detailed region of the edge.

Fig. 8. An example of the extraction process. (a) The full edge. (b) A detailed view of the same edge, with the line indicating piece contour, the circles showing the points used for secant calculation, and the *x*s representing the interior points to be sampled via interpolation. Axes show pixel coordinates for the shown piece.



(a) *landscape* (b) *construction*

Fig. 9. Similarity measures using formula (1) for three edges compared with all other edges, sorted by value. The similarity is here computed with $\alpha = 10$ to exaggerate the curve form, and the best similarity has the value 0.81. The graphs show three distinct groups: a few with high similarity, a substantial group with medium and a large group having no similarity.

Table 2
True positive rates for all edges matched against all other edges using different edge matching algorithms

Author	Puzzle (# pieces)		
	Construction (100)	Landscape (100)	Benjamin (54)
Kosiba et al. (1994)	64%	69%	76%
Chung et al. (1998)	58%	61%	81%
Yao and Shao (2003)	26%	51%	54%
Yao and Shao (2003) (HSI)	54%	75%	84%
Nielsen et al. (2006)	98%	100%	82%

algorithm. The results are shown in Table 2. Our method outperforms the other methods on the computer generated puzzles by a large margin. On real puzzles, however, its performance is close to that of the other methods and is slightly outperformed by the Yao et al. method used on the HSI components.

3. Solving puzzles

The algorithm used for solving puzzles is an adaptation of the ideas discussed in Wolfson et al. (1988) on solving puzzle borders, and Goldberg et al. (2004), who present a good method for solving interior puzzles using a completed border. The purpose of the algorithm is to verify the efficiency of our edge matching algorithm. The algorithm is described in Section 3.1 and is based on a greedy approach, that is, using the metaheuristic of choosing the next piece as the best local match and assuming that the series of such choices will result in a globally optimal solution. As such it relies heavily on accurate edge matching methods, and the chance of failure is likely to rapidly increase for very large puzzles.

We assume that all puzzle pieces are already segmented and the edge strips extracted. In Section 4, we shall see how this is achieved with real puzzle pieces.

3.1. Algorithm based on simple assembly heuristics

Current puzzle solvers often solve the border separately before assembling the interior of the puzzle. The reason for this is that solving the border is less complex than solving the interior because the number of border pieces is significantly smaller than the number of interior pieces (except for very small puzzles), and because each border piece is connected to exactly two other border pieces.

The problem of assembling the border such that a maximum confidence in the correctness is achieved can be described as a traveling salesman problem (TSP). The *cost* of going from city A to another city B is the equivalent of the *similarity* between pieces A and B. The solution to the border is the optimal way of fitting together the pieces $A \rightarrow B \rightarrow \cdots \rightarrow A$ to form the entire border of the puzzle.

Obtaining a solution in polynomial time may be done by a heuristic approach by solving the assignment problem instead, as first proposed by Wolfson et al. (1988). Despite the possibility of a solution different from the TSP, it is easily verified by checking that a single rectangular border is formed. If we use a good edge matching algorithm for the cost matrix the heuristic will often succeed. Should it fail, the heuristic may return two or more separate "tours" or orderings of border pieces. These tours, however, may be merged optimally in polynomial time as long as the number of tours is limited. Finally we note that the solution may be confirmed by the correct placements of the corner pieces.

Algorithm 1. Steps of the *Singlepiece* algorithm.

(1) Solve the puzzle border using the assignment problem heuristic.
- If more than one "tour" is found, merge them by testing all (valid) combinations.

(2) Repeat until all pieces have been placed:
   (a) Find all *pockets*, i.e. unoccupied positions with two adjacent pieces.
   (b) For each pocket place the best matching piece using the matching algorithm and similarity function $Sim(\ldots)$ described in Eq. (1).

Goldberg et al. (2004) suggested using a greedy approach to solve the interior. First, we identify *pockets*, which are unoccupied positions with at least two adjacent pieces already placed. When the border has already been completed, we start with four pockets. Until the interior is completed, there will always be at least one pocket. By always having at least two edges to compare to, the edge matching becomes more reliable, making the greedy selection of the best local match for each position a viable solution.

Our proposed "singlepiece" approach is shown as Algorithm 1.

3.2. Puzzle solver results

The final goal is of course to assemble a puzzle correctly (assuming that we know the true picture), and we measure our success by counting the incorrectly placed pieces when the algorithm halts. Note that since the algorithm will always place all pieces, errors in the early part of the process will cause a cascade of incorrect placements.

For our tests of the puzzle solver, we used puzzles with both rectangular and classical non-rectangular pieces. The results of these tests are shown in Table 3. In addition we have managed to solve two real puzzles of 24 and 54 pieces. Fig. 10 shows part of the solution found by our algorithm for the 54-piece puzzle.

Table 3
Percentage of incorrectly placed pieces for different subdivisions of `landscape` and `construction` using both rectangular and classical pieces

Size	Landscape		Construction	
	Rectangular	Classical	Rectangular	Classical
56 (7×8)	0%	0%	0%	0%
100 (10×10)	0%	0%	0%	0%
320 (16×20)	0%	0%	5,9%	7,2%
504 (18×28)	11%	–	–	–

A dash (–) indicates that no test result is available.

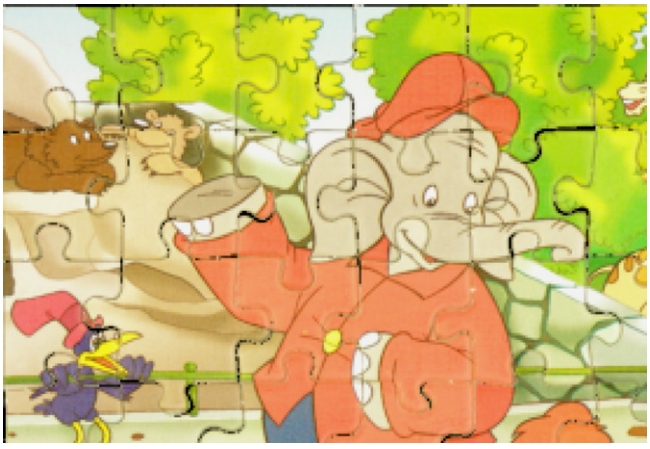


Fig. 10. Part of the solved 54-piece puzzle "Benjamin".

Table 4
Comparison with earlier methods

Author	Edge matching	Assembly	Piece count
Wolfson et al. (1988)	Shape	BH + ST	2 × 104
Webster et al. (1991)	Shape	ST	24
Burdea and Wolfson (1989)	Shape	BH + ST	"Few"
Kosiba et al. (1994)	Image + Shape	G	54[a]
Chung et al. (1998)	Image + Shape	BH + ST	54
Yao and Shao (2003)	Image + Shape	ST	"Dozens"
Goldberg et al. (2004)	Shape	BH + G	200
Singlepiece	Image	BH + G	54 (320)

In the Assembly column, BH is border heuristic, ST is search tree, and G is a greedy approach. Piece count specifies the largest puzzle solved without errors. Brackets indicate that computer generated pieces were used.

[a] Author indicates a "satisfactory" result was achieved.

In Table 4, we have presented an overview of previous puzzle solvers, which principles they use, and the size of the largest puzzles they reported were solved. We have appended our own algorithm for comparison.

4. Using real puzzle pieces

In order to demonstrate the efficiency of our edge matching algorithm in practice, we have tested our method on two classic puzzles of 24 and 54 pieces. During the digitization of these pieces we discovered a number of difficulties.

When using a scanner with the light source placed at an angle with respect to the photo sensors we observed a shadow on the background making segmentation more difficult. A larger problem, however, was that the piece itself had a shadow along one edge and a highlighted area along another. This is caused by the slight curvature created when the pieces are cut. The glossy surface of the piece has a specular reflection with the color of the light source, and has a weaker diffuse reflection at the edge farthest away. This problem is illustrated in Figs. 11 and 12 for two matching edge strips. These problems involve not only edge pixels but also the interior pixels within the region of the piece that we propose using for edge matching.

The pieces have been digitized using a digital camera in an environment where we can control the lighting. To avoid specular reflections in the surface we use a diffuse background lighting

Fig. 12. Example of strips from two matching edges. Shadows and highlights are visible close to the edge.

and no camera flash. To minimize problems with perspective, we only digitize one piece at a time at a fixed distance. Fig. 13 shows a digitized and segmented piece. Two complete sets of pieces (24 and 54 pieces) can be found at the (DIKU Image Group's FTP server, 2008).

A problem encountered when using a digital camera is that the camera has automatic brightness control. This means that pieces which contain dark areas are made brighter relative to pieces containing light areas. This is not a problem when using the hue color channel alone, as long as the camera does not saturate the RGB channels. Ideally the automatic brightness control should be disabled.

A final problem is the quality with which they were originally printed. In our images we noticed a distinct halftone pattern used in the printing process due to the limited color palette. As a result,

Fig. 13. Example of a photographed piece after segmentation.

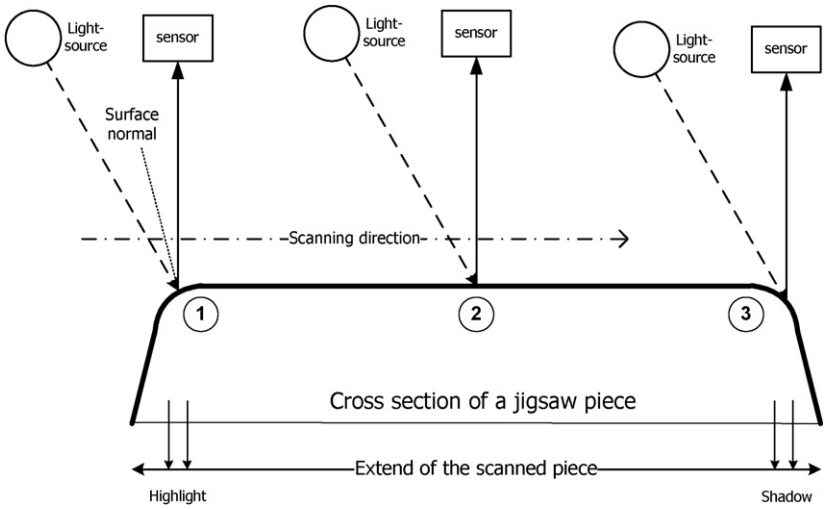
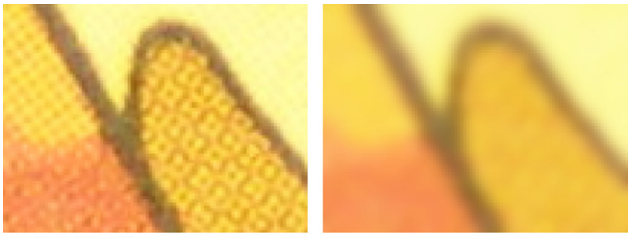


Fig. 11. Scanning with a single light source placed to the left of the photo sensor; the geometry of the cross-section is exaggerated. There are specular reflections at point 1 having the color of the light source. Point 2 shows the normal diffuse reflection. At point 3 the angle between the surface normal and the light source results in a poorly lit shadow.



(a) Halftone pattern. (b) Smoothed halftone pattern.

Fig. 14. (a) Example of the halftone pattern in a photographed piece. (b) Gaussian smoothed halftone pattern.

the pixel-wise comparison used for our edge matching will no longer be accurate, since the effects of dithering result in edge artifacts. We compensate for this problem by applying a Gaussian smoothing filter (with a standard deviation of 3 pixels for 150×150 pieces). Fig. 14 illustrates this application.

5. Topics for future work

What exactly constitutes a correct full or partial solution is a difficult question because we cannot avoid involving the human perception when judging the correctness. A human examines the solution visually to decide if it has misplaced pieces, but automating this is difficult as we cannot assume that we have a reference image. If we use a backtracking algorithm, we need to develop some heuristics for deciding success or failure when we try matching a piece to a partial solution. A simple solution is to stop if the similarity measure gets below a threshold. A correct solution is then a solution for which all similarities are "good enough". A better method would be to consider image features which are invariant for larger regions in the image and declare a possible failure when selecting a candidate piece which has significantly differing features, or in other words belongs to another class of pieces.

Algorithm 2. Steps of the *Superpiece* algorithm.

(1) Solve the puzzle border using the assignment problem heuristic.
(2) Classify all non-border pieces as described in Section 5.1. All non-classified pieces are placed in a *remainder* group and are not used for constructing superpieces.
(3) For each class, use some algorithm to assemble a superpiece. Discard pieces with similarity less than a given threshold by moving it to the remainder group.
(4) Repeat until there is only one superpiece left, i.e., the solution:
    (a) Compare each superpiece pair using average $Sim(\ldots)$ (from Section 2.1) for all connected edges and if their similarity is above a given threshold, combine them into a new super piece. Repeat until no more superpieces are combined.
    (b) Place a piece from the remainder group, where it matches best.
    (c) If there are no more pieces in the remainder group, one of the superpieces is dismantled and the pieces are placed in the remainder group.

Examining the solution space by finding the best matching piece in step 2.2 of the algorithm *singlepiece* is done by trying all pieces not yet placed in an arbitrary order. The algorithm is fast at solving the interior but it can easily fail for large puzzles, and since the number of combinations of puzzle pieces increase super-exponentially with the number of pieces as $(O(n!))$,[3] any form of solution based on searching through all potential solutions does in any case not scale well.

If, however, we are able to exploit the divide-and-conquer paradigm and divide the puzzle in such a way that each of the parts can be solved separately as a smaller puzzle, we will be able to decrease the theoretical running time significantly. This approach is inspired by the way humans solve large jigsaw puzzles by sorting the pieces by color, texture and shape and trying to assemble similar pieces into complete sub-puzzles. Again we need to form groups of related pieces.

Collections of matched and assembled pieces are called *superpieces*. The border of the puzzle can be assembled using the TSP heuristic as has been described in previous work, and can thus be considered one superpiece; the completed puzzle is another superpiece. Solving the puzzle by combining superpieces is, however, a challenge. Since the superpieces are not guaranteed to have pockets, a greedy approach will be much less accurate. Also, the shape of the superpieces will be quite irregular.

We have not implemented a working solution to the problem, but propose using the search algorithm *Superpiece* shown as Algorithm 2.

5.1. Classification

Classification of the pieces is not relevant for the selection part of Algorithm 1, but will be essential for any branch-and-bound algorithm and useful for deciding the correctness of a solution. A classification of a piece may be based on local features derived from the shape and the values of edge strip pixels, and on globally related features like color and texture of the images. This may be generalized to superpieces.

The shape of a piece does not generally indicate to which part of the puzzle it belongs. This leaves the images on the pieces as the main indicator, assuming that the picture formed by the assembled puzzle actually allows this. Selecting the best image features is a challenge, and we have made a preliminary investigation into color and texture features but have left shape as a future work item. We restrict our method to be used on "natural" images and not on random images.

Ideally, the number of classes should be determined automatically by the algorithm. However, this is far from trivial, and for the sake of simplicity we have chosen to use a fixed value which depends on the puzzle size. We expect that larger puzzles usually contain more classes, but that the number of classes only increase slowly with puzzle size. Therefore, we have made a qualitative assessment of a few puzzles and found by experimentation that $\sqrt[4]{n}+1$ is a reasonable number of classes to use, where n is the number of puzzle pieces.

5.2. Image features

An obvious image feature to use is the color of the piece and we have decided to use the HSI color space and compute the mean and variance for each of the three color channels giving a total of 6 features.[4]

If the puzzle is very large, the pieces will be small relative to the puzzle picture, and the color of a significant number of inter-con-

[3] Each piece has 4 sides, so with all permutations we get a total of $O(4^n n!)$ combinations.

[4] We need color components that are statistically independent. It is not necessary to use a calibrated color system like the CIE L*a*b* system because all images of one puzzle are generated in the same way.

nected pieces will likely be similar. Therefore we will also experiment with using texture features in our classification.

There are many ways to describe textures in images, e.g., co-occurrence matrices, high-order statistics, Markov random fields, wavelets or texels. We have chosen a set of statistics of co-occurrence matrices as texture descriptors, based on the assumption that similar textures are likely to have similar frequencies of pixel value pairs. After computing a co-occurrence matrix for each piece, we can derive a set of features. We chose the following features for both vertically and horizontally neighboring pixel pairs: contrast, C; inverse difference moment, *Idf*; and correlation *Corr* (Sonka et al., 1999; Bevk and Kononenko, 2002).

Assuming that the co-occurrence matrices are normalized:

$$\sum_{a,b} P_{\phi,d}(a,b) = 1. \tag{3}$$

The mentioned features are defined as

$$C = \sum_{a,b} |a-b|^{\kappa} P^{\lambda}_{\phi,d}(a,b) \tag{4}$$

$$Idf = \sum_{a,b;a\neq b} \frac{P^{\lambda}_{\phi,d}(a,b)}{|a-b|^{\kappa}}, \tag{5}$$

where we use typical values for $\kappa = 2$ and $\lambda = 1$.

$$\mathrm{Corr} = \frac{\sum_{a,b}(a-\mu_a)(b-\mu_b)P_{\phi,d}(a,b)}{\sigma_a \sigma_b}, \tag{6}$$

where μ_a, μ_b, σ_a, and σ_b is:

$$\mu_a = \sum_a a \sum_b P_{\phi,d}(a,b), \tag{7}$$

$$\mu_b = \sum_b b \sum_a P_{\phi,d}(a,b), \tag{8}$$

$$\sigma_a = \sqrt{\sum_a (a-\mu_a)^2 \sum_b P_{\phi,d}(a,b)}, \tag{9}$$

$$\sigma_b = \sqrt{\sum_b (b-\mu_b)^2 \sum_a P_{\phi,d}(a,b)}. \tag{10}$$

To reduce the directional dependency of these values, we compute the co-occurrence matrices with two orthogonal vectors $d_1 = [1,0]^{\mathrm{T}}$ and $d_2 = [0,1]^{\mathrm{T}}$. We use a length of one to detect fine detail as observed in, for instance, grassy or cloudy texture.

In summary, we are testing six color features (two per color channel), and six texture related features (three features in two directions using intensity pairs).

5.3. Classification results

The primary success criteria of the classification with a given set of classification features is to achieve a grouping of pieces, so that each group contains pieces which are likely to be inter-connected.

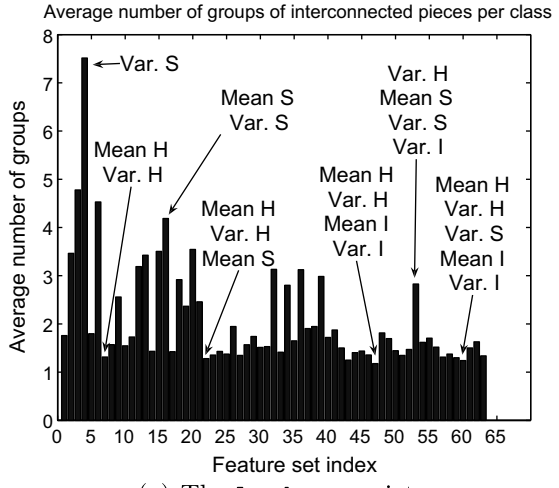
In order to find the best feature sets for the two images `landscape` and `construction`, we had to try all combinations since we have no model for the correlations between features, but we decided to test color and texture features separately to reduce the total number of tests.

During the tests, classes may be discarded due to an insufficient number of members to be able to compute reasonable Mahalanobis distances, so we have to make sure that the classification does not achieve a high degree of connectedness simply by discarding a significant number of classes. Thus the number of discarded classes is also recorded. Since we use a cut-off point for when a piece does not belong to a class in the clustering algorithm, we also need to record the percentage of unclassified pieces. To determine this cut-off point, we use Mahalanobis distance in the feature space.

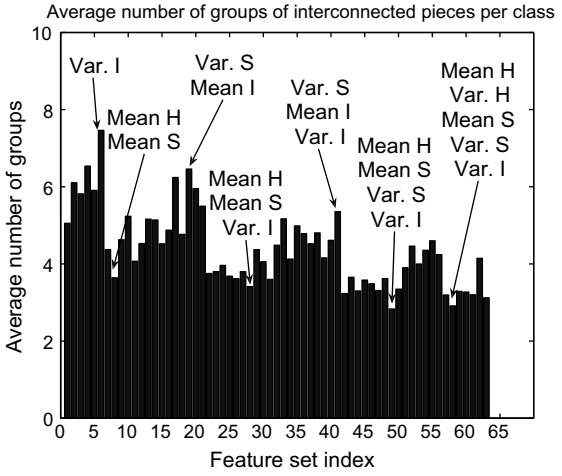
While the number of discarded classes and the classification ratio are relevant, they are not essential for the success of the classification. We simply have to verify that the values fall within acceptable levels. As the classification method determines a local minimum, we use a statistical average of the results over a thousand iterations.

Fig. 15 shows the average number of groups of inter-connected pieces per class for each (sub)set of color features. A perfect classification would result in one inter-connected group per class and at the same time not show dropping of classes or a significant number of unclassified pieces. As is evident from the two figures, the `landscape` puzzle is easier to classify with color features than the `construction` puzzle is. For the `landscape` puzzle, many feature sets are suitable for classification with only a handful being very poor, while for the `construction` puzzle color features alone are not particularly accurate.

The number of discarded classes is not shown in the figures. The best cases are only discarding 10 or 20 classes out of a thousand iterations and the worst about 100 classes. There does seem to be a small correlation between the size of feature sets and the number of dropped classes, but this may be due to the number of required members of the class to compute the Mahalanobis



(a) The `landscape` picture.



(b) The `construction` picture.

Fig. 15. Average number of groups of inter-connected pieces per class for each color (sub)set of features shown for both the used puzzle pictures. Several particularly high or low results are annotated with the features in question.

distance (dimension + 1), which increase with number of features. The classification ratio is satisfactory for all feature sets and cannot be used to discriminate between the investigated sets.

Using only texture features for classification proved to be generally less accurate than using only color features. The average number of inter-connected groups of pieces was found to be close to three. Combining the best feature sets found from the previous tests, we lost accuracy compared to only using color features, but gained accuracy compared to using only texture features. However, the loss was more significant when classifying the `landscape` puzzle than when classifying the `construction` puzzle.

The conclusion is that the feature sets used in connection with a given picture should be selected after an analysis of the class of pictures it belongs to.

6. Conclusion

We have developed a new method for edge matching using only image features, making them independent of the border shape of the puzzle pieces. Our *singlepiece* algorithm is the first successful attempt at solving jigsaw puzzles without using shape information. On computer generated puzzles, we were able to solve puzzles of up to 320 pieces – larger than the record of 200 held by Goldberg et al. (2004), but with which it cannot be directly compared. In addition we managed to solve a real puzzle of 54 pieces, which is on par with other algorithms using image information.

As such the image based edge matching has shown significant promise in the field of automatic puzzle solving.

We have furthermore investigated a new approach to puzzle solving where pieces can be classified according to image features and assembled in a number of smaller subpuzzles. This approach is described as the *superpiece* algorithm. The tests show that the classification of pieces based on image features is a distinct possibility for at least some puzzles, and we believe that this merits further investigation. Using an efficient classification method and assuming that an efficient solution using superpieces can be found we surmise that very large puzzles could be solved.

References

Bevk, M., Kononenko, I., 2002. A statistical approach to texture description of medical images: A preliminary study. In: Proceedings of the 15th IEEE Symposium on Computer-Based Medical Systems, CBMS 2002, pp. 239– 244.

Burdea, G.C., Wolfson, H.J., 1989. Solving jigsaw puzzles by a robot. IEEE Trans. Robotic. Autom. 5 (6), 752–764.

Chung, M., Fleck, M., Forsyth, D., 1998. Jigsaw puzzle solver using shape and color. In: Proceedings of the Fourth International Conference on Signal Processing ICSP'98, vol. 2, pp. 877–880.

DIKU Image Group's FTP server <ftp://ftp.diku.dk/diku/image/images/JigsawData.tgz> (accessed on 21 May 2008).

Freeman, H., Gardner, L., 1964. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. IEEE Trans. Electron. Comput. 13, 118–127.

Goldberg, D., Malon, C., Bern, M., 2004. A global approach to automatic solution of jigsaw puzzles. Comput. Geom. Theory Appl. 28 (2–3), 165–174.

Kosiba, D., Devaux, P., Balasubramanian, S., Gandhi, T., Kasturi, R., 1994. An automatic jigsaw puzzle solver. In: Proceedings 12th IAPR International Conference on Computer Vision and Image Processing, Jerusalem, October 1994, vol. 1, pp. 616–618.

Nielsen, T.R., Drewsen, P., Gütle, H., 2006. Løsning af puslespil ved hjælp af tekstur. graduate paper at Faculty of Computer Science, University of Copenhagen.

Otsu, N., 1979. A threshold selection method from gray level histograms. IEEE Trans. Syst. Man Cybernet. 9, 62–63.

Radack, G., Badler, N., 1982. Jigsaw puzzle matching using a boundary-centered polar encoding. Comput. Graphics Image Process. 19, 1–17.

Sonka, M., Hlavac, V., Boyle, R., 1999. Image Processing, Analysis, and Machine Vision, second ed. Brooks/Cole Publishing Company, Belmont, CA. ISBN 0-534-95393-X.

Webster, R.W., LaFollette, P.S., Stafford, R.L., 1991. Isthmus critical points for solving jigsaw puzzles in computer vision. IEEE Trans. Syst. Man Cybernet. 21, 1271–1278.

Wolfson, H., Schonberg, E., Kalvin, A., Lamdan, Y., 1988. Solving jigsaw puzzles by computer. Ann. Oper. Res. 12 (1–4), 51–64.

Yao, F.-H., Shao, G.-F., 2003. A shape and image merging technique to solve jigsaw puzzles. Pattern Recognition Lett. 24 (12), 1819–1835.