servation that the selection of the kernel covariance is very sensitive to the results.

Although we know how to select the optimal kernel size h and the threshold t of the classifier, the selection of the kernel function and the kernel covariance matrix is not clearly understood. For Gaussian distributions, the Gaussian kernel with the sample covariance matrix works very well, provided that a large number of samples is used to estimate the covariance. Unfortunately, in non-Gaussian cases, the sample covariance matrix does not reflect the local structure of the distribution, producing poor experimental results. If we could estimate the local covariance accurately, the Parzen density estimate would provide a good estimate of the Bayes error, and we could design the reduced Parzen classifier with a small number of representatives.

IV. Summary

An algorithm was proposed to select a subset of representative samples from a given data set which preserves the Parzen density estimate. If an approximate Bayes classifier is designed using these representatives, nearly optimal discrimination is achieved, even for a significantly reduced number of representatives.

References

[1] E. Parzen, "An estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, pp. 1065–1076, 1962.

[2] D. J. Hand, *Kernel Discriminant Analysis*. Chichester, England: Research Studies, 1982.

[3] P. A. Devijver and J. Kittler, "On the edited nearest neighbor rule," in *Proc. 5th Int. Conf. Pattern Recognition*, 1980, pp. 72–80.

[4] G. W. Gates, "The reduced nearest neighbor rule," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 431–433, 1972.

[5] P. E. Hart, "The condensed nearest neighbor rule," *IEEE Trans. Inform. Theory*, vol. IT-14, pp. 515–516, 1968.

[6] S. Geman and C. R. Hwang, "Nonparametric maximum likelihood estimation by the method of series," *Ann. Statist.*, vol. 10, pp. 401–414, 1982.

[7] B. S. Everitt and D. J. Hand, *Finite Mixture Distributions*. London:Chapman and Hall, 1981.

[8] D. M. Titterington, A. F. M. Smith, and U. E. Markov, *Statistical Analysis of Finite Mixture Distributions*. New York: Wiley, 1985.

[9] K. Fukunaga and J. M. Mantock, "Nonparametric data reduction," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 115–118, 1984.

[10] K. Fukunaga and D. M. Hummels, "Bayes error estimation using Parzen and k-NN procedures," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, pp. 634–643, 1987.

An Efficient Uniform Cost Algorithm Applied to Distance Transforms

BEN J. H. VERWER, PIET W. VERBEEK, AND SIMON T. DEKKER

***Abstract*—In artificial intelligence, a number of search algorithms is available for finding shortest paths in graphs. The uniform cost algorithm is a special case of one of those algorithms, the A^*-algorithm. In the uniform cost algorithm, nodes are expanded in order of increasing cost. We have developed an efficient version of this algorithm for integer cost values. Nodes are sorted by storing them at predefined places (bucket sort), keeping the overhead low. The algorithm is applied to general distance transformation. A constrained distance transform is an operation which calculates at each pixel of an image the distance to the nearest pixel of a reference set, distance being defined as minimum path length. The uniform cost algorithm, in the constrained case, proves to be the best solution for distance transformation. It is fast, the processing time is independent of the complexity of the image and memory requirements are moderate.**

***Index Terms*—A^*-algorithm, bucket sort, distance transform, uniform cost propagation.**

I. Introduction

Heuristic search is widely used in optimization theory, problem solving, games theory, and path finding. A^* is a basic algorithm in heuristic search.

A^* finds the cheapest path in a graph from a start node to the closest goal node [1]. Nodes are processed in a specific order, determined by an heuristic evaluation function $f(n)$, defined as an estimate of the cost of the path from the start node to the closest goal node constrained to go through node n. The function $f(n)$ is the sum of two functions $g(n)$ and $h(n)$, $g(n)$ representing the costs from the start node to node n and $h(n)$ an estimate of the costs from node n to the goal node. In A^*, $g(n)$ is calculated recursively during the search process: if n' is a successor node of n, then $g(n')$ is the sum of $g(n)$ and the cost $c(n, n')$ associated with the arc between n and n'. The estimate $h(n)$ should be based on the location of the goal and characteristics of the problem domain.

A^* can be described in four steps [1].

1) Mark the start node s "open," calculate $f(s)$.

2) Select the open node n whose f value is smallest. If ties occur, resolve in favor of a goal node.

3) If n is a goal node, mark n "closed" and terminate the algorithm.

4) Otherwise, mark n "closed," calculate f for each successor node of n and mark "open" each successor not already marked "closed." Remark as "open" any closed node n_i which is a successor of n for which $f(n_i)$ is smaller now than it was when n_i was marked "closed." Goto step 2.

A special case of A^* is the uniform cost algorithm where heuristic information is not used: $h(n)$ is put to 0. We propose an efficient method to implement this algorithm for integer cost values. Our approach is useful if goal-oriented behavior is not of interest or if the loss of time due to the calculation of $h(n)$ exceeds the gain introduced by the number of nodes which need not be processed.

II. Efficient Uniform Cost Propogation

We propose to use a bucket structure to implement the node labeling. If a node is labeled "open" it is stored in a bucket. Each bucket has associated with it a particular cost value. In bucket i all nodes with cost value $g(n) = i$ are stored. A cumbersome sorting procedure to find the node with the lowest cost value is avoided by emptying the buckets in order of increasing cost.

The efficient uniform cost propagation we propose starts by assigning the cost 0 to the start nodes and the value infinite to the other nodes (or a cost larger than the maximum cost to occur). So, the start nodes are stored in bucket 0.

The buckets are then emptied in order of increasing cost, starting with bucket 0. A node retrieved from a bucket checks whether it can propagate costs to adjacent nodes. An adjacent node receives the cost of the generating node plus the cost of the arc between them, if the adjacent node's original cost was higher than the newly calculated cost. In that case, the node is stored in the corresponding bucket to be retrieved at the proper time. After having checked all

Manuscript received June 19, 1987; revised February 8, 1988. This work was supported by the Dutch Government as part of the SPIN-FLAIR-2 "Delft Intelligent Assembly Cel" project.

The authors are with the Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, Lorentzweg 1, 2625 CJ Delft, The Netherlands.

IEEE Log Number 8825725.

0162-8828/89/0400-0425$01.00 © 1989 IEEE

adjacent nodes, the generating node is removed from the bucket (labeled "closed").

In [1] it is proven that the costs of all closed nodes are correct if $h(n)$ has a property called *consistency*. An equivalent condition is *monotonicity* [2]. An heuristic function is called monotone if $h(n) \leq c(n, n') + h(n')$. If no heuristic information is used ($h(n) = h(n') = 0$) and if negative arc costs are not allowed ($c(n, n') \geq 0$) then the heuristic function $h(n)$ is certainly monotone and then the cost of a generating node will always be correct.

The ordering of nodes inside a bucket is not relevant. Moreover, the buckets can circulate: if a bucket has been emptied, it can be reassigned to the next cost value that could be required. At any time, a limited number of buckets, equal to the maximum arc cost plus 1, is sufficient (nodes in the current bucket can only generate costs between the current cost and the current cost plus the maximum arc cost; buckets for the other cost values are superfluous). The buckets are thus addressed modulo the maximum arc cost plus 1.

The speed of the algorithm is due to the simple administration involved. A sorting of "open" nodes as in the more general A^*-algorithm is not required because nodes are stored at a predefined place. This "sorting" method is generally known under the name *bucket sort*. It is the only "sorting" method with a time complexity of $O(n)$ where n is the number of elements to be sorted.

The cost of a generating node is always correct (see above), the cost of a generated node not necessarily. If the cost of a generated node is not correct, it will be corrected before it propagates itself. At the time it is corrected, the node will remain in a wrong bucket. To prevent the node from being processed twice and to circumvent a cumbersome removal procedure, a comparison is included. Upon each retrieval of a node, the value of the node and the index of the bucket from which the node was retrieved, should match. If they do not match, the node has already been processed and is therefore treated as "closed."

III. Distance Transforms

A. Definitions

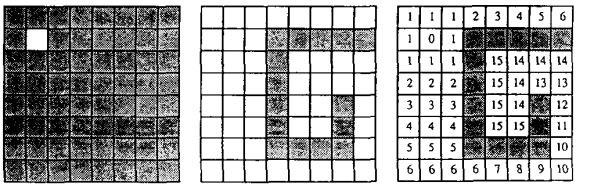
Distance transforms are a tool in image processing, with applications in skeletonization, medial axis transformation, convex hull extraction, clustering, matching, and robot path finding. A *constrained distance transform* is a transform of two binary images S and R to a gray-value image G. A gray value in G represents the minimum distance of an object pixel in S to the closest pixel in R, the reference image. The background pixels of S form the *constraint image*. Examples are given in Figs. 1, 2, and 3. Object pixels are shown white, background pixels black.

The traditional *distance transform* [3] is a special case of the constrained distance transform, in which the reference image R is the constraint image of S. The algorithm we present is only faster than existing distance transforms in the constrained case, although in some applications of the normal distance transform its usefulness could also lie in the processing of pixels in order of increasing distance (e.g., skeletonization [4]).

B. Metrics

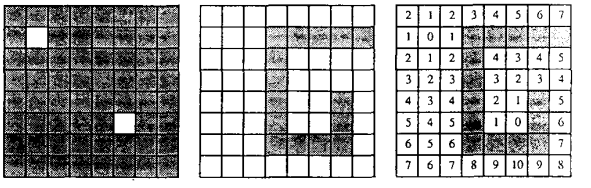
The distance d between two pixels is defined as the *minimum path length* of all the paths, entirely contained in S, between the two pixels. A *path* is defined as a sequence of basic vectors b_i, a *path length* as the sum of the distance values d_i associated with the basic vectors b_i; and d_i are called *chamfer* distances [5].

In N dimensions the elementary basic vectors are $(x_1, x_2, \cdots, x_N)^T$ with $x_i \in \{0, \pm 1\}$, $i = 1, 2, \cdots, N$ (see Fig. 4 for the 2-D elementary basic vectors). Nonelementary basic vectors have components outside the set $\{0, \pm 1\}$ [6]. In [7] it is proven that the extension to nonelementary basic vectors guarantees that a specified maximum relative error is reached. From hereon we will focus on the elementary basic vectors only, although the algorithm presented is equally applicable for nonelementary basic vectors.



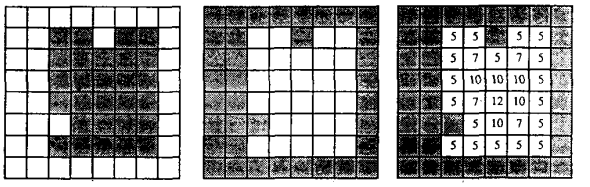
(a) (b) (c)

Fig. 1. Chessboard metric, one reference pixel. (a) Reference image R. (b) Constraint image, conjugate of S. (c) Distance image G.



(a) (b) (c)

Fig. 2. Chessboard metric, one reference pixel. (a) Reference image R. (b) Constraint image, conjugate of S. (c) Distance image G.



(a) (b) (c)

Fig. 3. Octagonal metric, background is reference set. (a) Reference image R. (b) Constraint image, conjugate of S. (c) Distance image G.

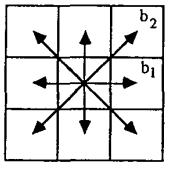


Fig. 4. Elementary basic vectors in 2 dimensions.

In two dimensions the maximum relative error can be minimized to 3.96 percent. In three dimensions to 1.36 percent [7]. For the associated (real valued) chamfer distances, we refer to Table I.

Two remarks:

- There is no limit on the dimensionality of the space. Robot path finding is performed in a six- or seven-dimensional state space [8].
- Other error measures than the maximum relative error are the maximum absolute error [5] and the mean square error [9]. For these error measures a general result as in [7] has not been established yet.

Integer approximations are also shown in Table I, from [10]. In integer approximations only the ratio is optimized, after which a scalar correction is sufficient to obtain absolute distances. The first integer approximation in Table I is the chessboard metric (e.g., Fig. 1), the second approximation the city-block distance (e.g., Fig. 2). As can be seen, these are not optimal metrics with respect to the Euclidean metric. The general 2-D coefficient pairs $\{d_1, d_2\}$ will be called an octagonal distance measure because the points at equal distance to one reference pixel form octagonals (e.g., Fig.

TABLE I
CHAMFER DISTANCES OF THE ELEMENTARY BASIC VECTORS IN 2 AND 3 DIMENSIONS, PLUS THE REQUIRED SCALING FACTOR AND THE MAXIMUM RELATIVE ERROR [10]

	d_1	d_2	d_3	scaling	error (%)
2-D best set (real valued)	0.9604	1.3583		1.00	3.96
2-D integer approximations	1	1		0.85	17.16
	1	2		1.21	17.16
	2	3		2.12	5.57
	5	7		5.17	4.21
	12	17		12.50	4.00
	29	41		30.19	3.96
3-D best set (real valued)	0.9398	1.3291	1.6278	1.00	1.36
3-D integer approximations	1	1	1	0.79	26.80
	1	2	3	1.27	26.80
	3	4	5	3.07	7.95
	4	6	7	4.29	6.79
	7	10	12	7.40	6.39
	11	16	19	11.71	6.32
	12	17	21	12.80	6.26
	19	26	33	20.24	6.11

3). We propose to use the same name in higher dimension, although the points at equal distance then form other shapes.

C. Algorithms

Borgefors [5], [6] uses a filter approach to calculate at each position in an object the distance to its contour. Two passes suffice. Dorst and Verbeek [11] use the same approach for the constrained case. The image now has to be processed until convergence. Piper and Granum [12] tested several combinations of scanning methods and masks. They propose some improved sequential transforms, but also show that for some applications an ordered propagation algorithm is better suited. They propagate distances from the reference points. The adopted connectivity determines the order in which pixels are processed. Pixels are thus processed in order of increasing chessboard distance, while the distances calculated are octagonal. This technique is called breadth-first in heuristic search. In some cases (see Fig. 5) this leads to major recalculations of distance values. Therefore, Piper and Granum decide to change the order of calculation in such cases. An ad hoc solution. We believe that the uniform cost algorithm more closely corresponds to the concept of propagating distances.

We will not compare our algorithm to parallel algorithms. The parallel distance transforms will have to do as many iterations as the largest distance in the image, while our algorithm needs to process each pixel only once. Any software implementation will therefore be in favor of our algorithms. For special purpose hardware, the same holds. Only if one has general purpose parallel hardware at one's disposal, which does not allow the manipulation of addresses, the balance could shift in favor of the parallel algorithms or in favor of the sequential local transforms. Any comparison in that case is useless.

IV. THE UNIFORM COST ALGORITHM APPLIED TO DISTANCE TRANSFORMS

Applied to distance transforms, the uniform cost algorithm operates in an image. The nodes are the pixels. The arcs are determined by the basic vectors of the metric. Constraint pixels are labeled "write-protected" and do not participate in the process. (If nonelementary basic vectors are used, care has to be taken that basic vectors do not cross constraints. In that case the pixels crossed by the nonelementary basic vector have to be nonconstraint pixels as well.) The algorithm described in Section III can be implemented straightforward. The chamfer distances d_i satisfy the condition that $c(n, n') \geq 0$. After having generated distances a pixel is labelled "closed." Each "closed" pixel has the correct distance value and is processed only once.

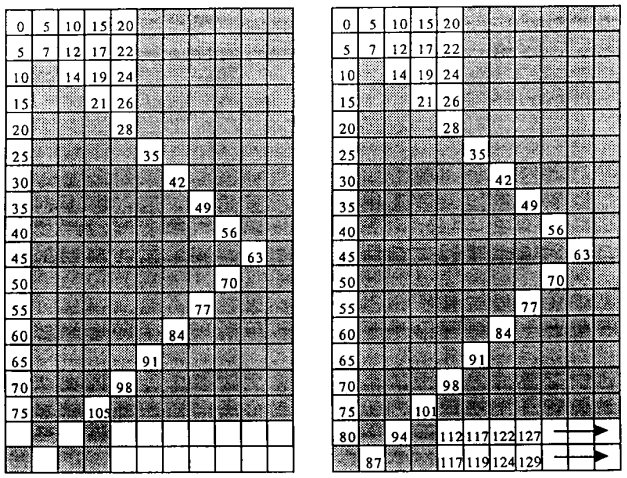


Fig. 5. The ordered propagated distance transform encounters an obstacle which divides the wavefront in two; all points after the obstacles will generate distances twice. An ad hoc solution for this problem is required. (a) Situation after 15 generations. (b) Situation after 19 generations, distance 101 will have to adjust distance 112 to 108, etc.

In the Sections VI and VII, we will compare the performance of our algorithm to the traditional algorithm. First we will, however, discuss a difference between the approach of the A^*-algorithm and the approach of distance transform algorithms.

V. WRITE VERSUS READ FORMALISM

There is a subtle difference in the approach of our algorithm versus other distance transforms. In Borgefors [5], [6], Dorst and Verbeek [11], and Piper and Granum [12], at each pixel a new value was calculated by

$$p_0 = \min \{ p_i + c_i \} \quad \text{"read formalism"} \tag{1}$$

in which p_0 is the value of the central pixel, in which the p_i are the values of the neighboring pixels, including the central pixel, and in which the c_i are the corresponding chamfer distances. This approach clearly shows the influence of image processing. The operation mimics a convolution. Coefficients are added to instead of multiplied with the pixel values and a minimization replaces the summation.

In our algorithm the minimization disappears and individual comparisons takeover

$$\forall_i: \; if(p_0 + c_i < p_i)\, p_i = p_0 + c_i \quad \text{"write formalism."} \tag{2}$$

Of course the computational complexity is the same, the comparisons in formula (2) correspond to the intermediate calculations in evaluating formula (1).

In distance transforms both formalism are allowed (an example of a uniform cost algorithm which uses the read-formalism is given in [4]).

VI. PROCESSING TIMES

The algorithms have been tested on a number of 256 × 256 test images. 3-D algorithms are available as well, but the images are more difficult to display here. The basic differences between the algorithms are the same.

The constraint images were respectively an empty image and the images as shown in Fig. 6(a)–(d). The reference image contained one pixel located at position (1, 1). The output of the distance

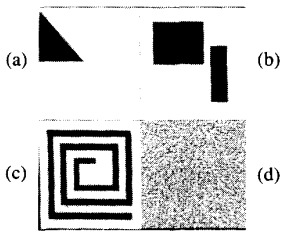


Fig. 6. Four of the five test images used in the comparison between the different distance transforms. A triangle (a), two simple blocks (b), a spiral (c), and a maze (d).

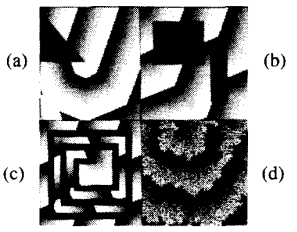


Fig. 7. The distance transformed test images. The reference pixel was in all 4 images located in the upper left corner of the image. The distances are shown modulo 256.

transforms, identical for all algorithms, is shown in Fig. 7(a)-(d). The distance values are displayed modulo 256 to show the pixels at equal distance more clearly. The metric used was the smallest integer approximation of the octagonal metric, the coefficient pair $\{2, 3\}$. This to avoid overflow in the image memory.

The processing times are printed in Table II. The experiments were performed on a 10 MHz 68000 processor running under UNIX. The algorithms were all written in C. As can be concluded from the table the processing times for empty constraint images are comparable. In that case, our algorithm does not perform better (even ± 20 percent worse than the sequential local distance transform).

As soon as obstacles appear, the sequential local transform of Dorst and Verbeek [5] takes much more time, linearly dependent on the required number of passes through the image. As shown in [12] the number of passes is very data-dependent. The performance can only deteriorate from the nonconstrained case.

The processing time of the ordered propogation distance transform [12] can increase or decrease with respect to the nonconstrained case. For the triangle of Fig. 6(a), the processing time jumps up due to the difference between the octagonal metric and the chessboard connectivity (as explained in Section III). In the spiral and the maze, this difference is not relevant and the processing times decrease because there are less distance values to be calculated.

The uniform cost algorithm accelerates if more obstacle pixels are present. The gain is not linearly proportional to the number of pixels which are obstacle pixels (see, e.g., the processing time of the maze) because some obstacle pixels are still addressed before it is noted that they need not be processed. The processing time is therefore dependent on the data, but obstacles can not slow the algorithm down!

Concluding, we believe the uniform cost algorithm is the most elegant solution for distance transforms in the constrained case. Other algorithms can be satisfactory in specific cases, but the uniform cost algorithm has a good performance at all times.

VII. Memory Requirements

An upper limit for the size of one bucket is the size of the image, if the whole image consists of reference points. In practical situations a smaller size will suffice. The order of the number of pixels which have to be stored is, except for in pathological cases [12], one order smaller than the order of the total number of pixels which are present.

If the dynamical allocation scheme is used, the size of the buckets is determined automatically. In a dynamical allocation scheme the buckets are built of chunks, each of which can contain a fixed number of pixels. If a chunk is filled during the calculations, the next free chunk is used or a new one is allocated. In Fig. 8, the allocated memory is shown as a function of the number of points (4 bytes) per chunk. The allocated memory shows in all cases a decaying exponential, followed by a sawtooth and finally by a linear function. The decaying exponential is due to overhead over the linked list structure, being huge for small chunksizes. The sawtooth is due to the quantization of the chunksize, one pointer extra per chunk can make a set of chunks superfluous. And finally, the linear function arises when the minimum number of chunks, equal to the number of buckets, is reached. A larger chunksize will not decrease the number of chunks, but only increase the allocated memory.

TABLE II
Processing Times in Seconds of the Uniform Cost Distance Transform (UCDT), the Ordered Propagated Distance Transform (OPDT) and the Sequential Local Distance Transform (SLDT). Also Indicated is the Percentage of Object Pixels of the Total Number of Pixels in the Image

Image	#object/#total pixels	UCDT	OPDT	SLDT
Empty image (except for edges)	1.6%	10.2	13.5	8.5
Triangle (figure 6a)	12.5%	8.9	35.4	11.4
2 simple blocks (figure 6b)	25.0%	7.6	10.3	9.9
Spiral (figure 6c)	37.5%	4.9	8.2	14.2
Maze (figure 6d)	50.0%	4.8	6.1	50.6

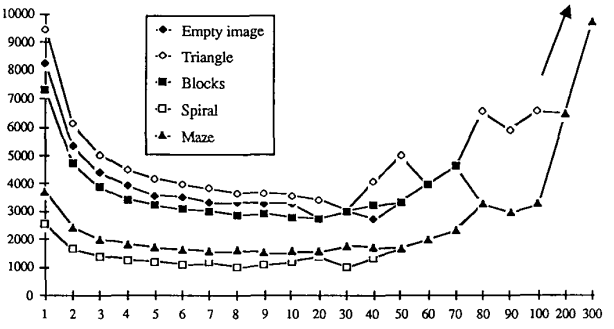


Fig. 8. Allocated memory (in bytes) as a function of chunksize (in number of pointers). The results for all five test images are shown.

Fig. 9 shows the influence of the chunksize on the processing time. The asymptotic behavior quickly stabilizes after ± 10. The time involved to switch from one chunk to the next is than negligible. It can be concluded that a reasonable chunksize is not data-dependent. Any number between 15 and 100 is satisfactory.

VIII. Conclusion

Our algorithm for uniform cost transforms efficiently sorts "open" nodes by putting them at a previously defined place. It can be used as a heuristic search technique in those cases where the use of heuristic information is not possible or time consuming.

Applied as distance transform the algorithm is as fast as other known algorithms. Applied as constrained distance transform the algorithm is generally faster than sequential local transforms. Memory requirements are moderate. We believe our algorithm is a more elegant solution and more easily programmable than the propagated distance transform. Each pixel is processed only once. The simple approach offers perspectives for hardware implementations.

Our algorithm can be used as a substitute for a filtering technique in image processing and as a simple heuristic search technique for artificial intelligence purposes.

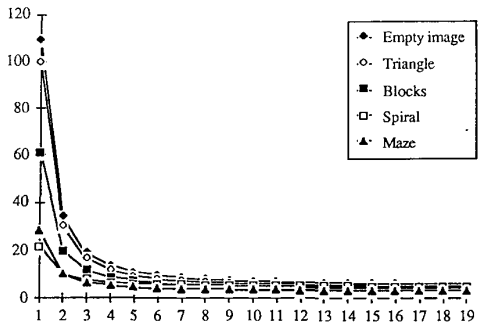


Fig. 9. Processing time (in seconds) as a function of chunksize (in number of pointers). The results for all five test images are given.

References

[1] P. E. Hart, N. J. Nilson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, pp. 100–107, Feb. 1968.
[2] J. Pearl, *Heuristics*. Reading, MA: Addison-Wesley, 1984.
[3] A. Rosenfeld and J. L. Pfaltz, "Distance functions in digital pictures," *Patt. Recog.*, vol. 1, no. 1, pp. 33–61, 1968.
[4] B. J. H. Verwer, "Improved metrics in image processing applied to the Hilditch skeleton," in *Proc. 9th Int. Conf. Pattern Recognition*, Rome, Italy, Nov. 14–17, 1988, pp. 137–142.
[5] G. Borgefors, "Distance transformations in arbitrary dimensions," *Comput. Vision, Graph. Image Process.*, vol. 27, pp. 321–345, 1984.
[6] ——, "Distance transformations in digital images," *Comput Vision, Graph. Image Process.*, vol. 34, pp. 344–371, 1986.
[7] M. Yamashita and T. Ibaraki, "Distances defined by neighborhood sequences," *Pattern Recog.*, vol. 19, no. 3, pp. 237–246, 1986.
[8] P. W. Verbeek, L. Dorst, B. J. H. Verwer, and F. C. A. Groen, "Collision avoidance and path finding through constrained distance transformation in robot state space," in *Proc. Intelligent Autonomous Systems*. Amsterdam, The Netherlands, Dec. 8–11, 1986, pp. 634–641.
[9] L. Dorst, "Discrete straight line segments: Parameters, primitives and properties," Ph.D. dissertation, Delft Univ. Technol., The Netherlands, 1986.
[10] S. T. Dekker, to be published.
[11] L. Dorst and P. W. Verbeek, "The constrained distance transformation: A pseudo-Euclidian, recursive implementation of the Lee-algorithm," in *Proc. European Signal Process. Conf. 1986*, The Hague, The Netherlands, Sept. 2–5, 1986, pp. 917–920.
[12] J. Piper and E. Granum, "Computing distance transformations in convex and non-convex domains," *Patt. Recog.*, vol. 20, no. 6, pp. 599–615, 1987.

Recognition of Handwritten Chinese Characters by Modified Hough Transform Techniques

FANG-HSUAN CHENG, WEN-HSING HSU, AND MEI-YING CHEN

***Abstract*—The Hough transform algorithm is a powerful technique for line detection. Its main advantages are that it is relatively unaffected by gaps in lines and by noise. This correspondence proposes a modified Hough transform (MHT) method for the recognition of handwritten Chinese characters. Chinese characters are mapped from the spatial domain into the parametric one for stroke extraction, and the dynamic programming matching (DP matching) algorithm is applied to recognize Chinese characters. This is a new approach to the application of the Hough transform and also a new attempt at the stroke extraction of handwritten Chinese characters. Two experiments had been conducted for a database called ETL8, which contains 881 Chinese characters and 160 variations for each one, to prove the usefulness of the MHT and DP matching methods, and an actual recognition rate of 94.5 percent is obtained for 351 Chinese characters in the ETL8.**

***Index Terms*—Accumulative matrix, dynamic programming matching, feature vector coding, modified Hough transform, peak detection, stroke extraction, weighted accumulative value.**

I. Introduction

Feature extraction is the most important part in pattern recognition because features are the main key to recognize the unknown object. In the context of feature extraction for automatic character recognition, features are customarily defined as two types: global and local features. The principle of global features is based on the transformation which can map the character matrix into a new domain to extract features. The selection of local features is based on geometrical and topological properties of the characters, such as stroke direction [1], stroke density [2], cellular feature [3], and stroke length and position of [4], [5], etc.

In the global feature approach, large storage and long computational time are necessary for its high complexity. Several orthogonal transformations such as Fourier [6], Walsh [7], and Karhunen–Loeve [8], [9] transforms have been suggested as a solution to this problem in which the pattern is transformed into a new domain and features that best represent the pattern are selected in that domain. Among them, the K–L expansion has been proved to be a successful algorithm in machine-printed Chinese character recognition; an experiment with a recognition rate of 94.6 percent was carried out by Kurosawa *et al.* [10] for the recognition of handwritten Chinese characters. However, the probems of time consumption and complex computation become fatal in this approach.

With regard to the local feature approach, it is well known as a powerful technique for the recognition of handwritten Chinese characters, but it is very sensitive to noise. The success of a recognition system taking such an approach is greatly influenced by how to extract the invariant features from the Chinese character correctly. This problem has been studied for a long time, and a lot of feature extraction methods which can be divided into four categories had been surveyed by Mori *et al.* [11]. They are stroke distribution [12], [13], stroke analysis [14]–[16], background feature distribution [17], [18], and a combination of the front ones. Among them, the stroke analysis is the most traditional approach which selects the "stroke" as the feature to describe the character in accordance with the construction of Chinese characters. Although each Chinese character is composed of a set of strokes, it is difficult for us to extract them correctly because they are very sensitive to noise. Therefore, how to extract the stable strokes becomes the main problem in this field.

In this correspondence, a new method, called the modified Hough transform (MHT) method, is proposed for stroke extraction of handwritten Chinese characters, and the dynamic programming matching method is applied to recognize Chinese characters. The MHT method transforms the Chinese character from the spatial domain into the parametric domain to extract strokes and treats these strokes as local ones for matching. This is a new approach to the application of the Hough transform, and also a new attempt at the stroke extraction of handwritten Chinese characters. There are two reasons for the motivation. The first one is that most of the strokes constituting Chinese characters are almost linear and can be detected by the HT as lines. The second one is that the HT method

Manuscript received August 27, 1986; revised August 20, 1988. Recommended for acceptance by C. Y. Suen.

F.-H. Cheng and M.-Y. Chen are with the Institute of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan 30043, Republic of China.

W.-S. Hsu is with the Institute of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan 30043, Republic of China, and the Institute of Information Science, Academia Sinica, Taipei. Taiwan 11529, Republic of China.

IEEE Log Number 8825640.