Recognizing Planar Symbols with Severe Perspective Deformation

Linlin Li, Chew Lim Tan
School of Computing, National University of Singapore
{lilinlin,tancl}@comp.nus.edu.sg

***Abstract*—A common problem encountered in recognizing symbols in real-scene images is the perspective deformation. In this paper, a recognition method resistant to perspective deformation is proposed. Particularly, a symbol is represented by a sequence of cross ratio spectra, in which perspective deformation is modeled as an uneven stretching deformation. Dynamic Time Warping algorithm is employed to compare the similarity of spectra and to find the pixel level correspondence between two symbols. The proposed method shows not only good resistance to severe perspective deformation and other common noise in real scene images, but also good discriminating ability to similar symbols.**

***Index Terms*—Symbol recognition, Perspective deformation, Dynamic time warping.**

I. Introduction

Planar symbols, such as characters and signs, are widely used to represent meanings in our daily life. With the advancement of camera technology, recognition of planar symbols with perspective deformation becomes an extremely important issue, as a foundation of many applications like traffic sign recognition, real-scene character recognition, and camera-based document image processing.

In order to resolve the problem of perspective deformation, a popular solution is to estimate the perspective transformation matrix and reconstruct the fronto-parallel view. Jelinek et al [6] proposed a rectification method making use of vanishing points to estimate the perspective projection matrix. This approach has been widely adopted when straight lines can be found in the document image. In camera-based document-image processing applications, text line, column edge, and stroke boundary were employed [16] [10]. In real-scene word recognition applications, quadrilateral edges of the text area were used [4]. However, the presence of these lines is highly Application dependent, and thus techniques designed particularly for one application may not be applicable to another. When no straight line can be found in an image, Orrite et al [15] suggested to use bitengent points to estimate the transformation matrix. Another solution is to approximate the perspective transformation by an affine transformation, and then remove the affine deformation by normalization [14] [22]. Since affine transformation matrix has less free parameters, it is easier to be evaluated. However, the approximation only holds when the distance between the object and the camera is much greater than the size of the object.

A more thorough solution is to recognize the symbol directly, and this is the very solution that we are aiming for. Approaches here can be divided into two categories, namely structure based and descriptor based. Structure-based approaches are popular for direct recognition. Many structure primitives are invariant to perspective deformation. In [11], structural invariants, namely, ascender and descender, vertical runs, and water reservoirs were employed to classify English characters into a reduced symbol set. Nevertheless, structural invariants alone are not discriminating enough to differentiate structurally similar shape, such as o and 0. Therefore algebraic invariants drawn from structural primitives are more frequently used than structure primitives. A model-based recognition system, called LEWIS [17], made use of algebraic invariants computed from primitive sets. Suk et al [18] derived a set of invariants called absolute projective triangular invariant from the vertices of polygons. However, as far as symbols are concerned, a drawback of these approaches is that desirable structure primitives are not always present. Recognition methods in the other category are based on shape descriptors. Such methods are structure-independent and thus are more generally applicable. MPEG-7 visual contour shape descriptors (CSS) [2], a global image descriptor based on curvature scale space, is widely employed in image retrieval applications where perspective deformation is involved. Scale Invariant Feature Transform (SIFT) [9] is a local Affine-invariant descriptor based on intensity information. A comparative evaluation of local descriptors in [12] showed that SIFT descriptor performs significantly better than many other local descriptors proposed in the literature. However, both of them were employed to identify complex objects with great variation in intensity previously. They may not be able to give distinctive descriptions of symbols with simple and symmetrical structures, which will be further illustrated in our experiment. Another important descriptor is Shape Context [1], which shows good ability to handle moderate perspective deformation in [12] and to discriminate symbols with non-rigid transformation in [1]. The last method, proposed by Suk [19], is a way to generate projective moment invariants with a form of infinite series. However, discriminating ability and convergence problem were left open, and was later challenged by [21].

In our early paper [8], a character recognition method was proposed, based on a descriptor named cross ratio spectrum. In this paper, the method is applied to identify planar symbols in a general case. In addition, a solution to the speed problem of the recognition method is also presented.

II. Cross Ratio Spectrum

Cross Ratio is a fundamental invariant for projective transformation [13]. The cross ratio of four collinear points (P_1, P_2, P_3, P_4) displayed in order is defined as:

$$cross\_ratio(P_1, P_2, P_3, P_4) = \frac{P_1P_3}{P_2P_3} / \frac{P_1P_4}{P_2P_4} \tag{1}$$

where P_iP_j denotes the distance between P_i and P_j. $cross\_ratio(P_1, P_2, P_3, P_4)$ remains constant under any projective transformation.

A. Cross Ratio Spectrum

Fig. 1 shows a character 'H' under a fronto-parallel view (H) and a perspective view (H'). Suppose pixels $P_1 \in H$ and $P_k \in H$ have mapping pixels $P'_1 \in H'$ and $P'_k \in H'$, respectively. Then I_1 and I_2 (intersections of the strokes and line P_1P_k) have mapping pixels I'_1 and I'_2 (intersections of

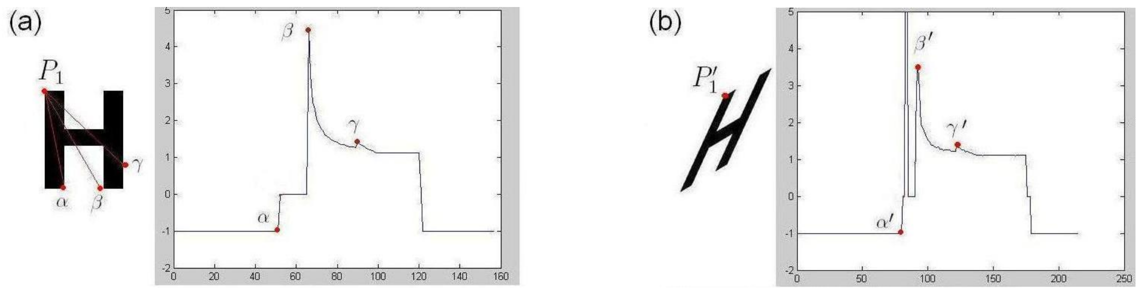

Fig. 2. CRSs of mapping pixels P_1, and P_1'.

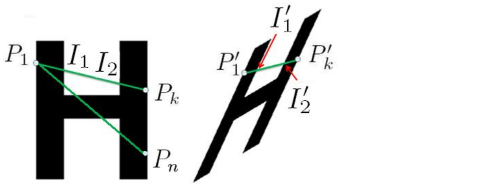

Fig. 1. Character 'H' in the fronto-parallel view and a perspective view.

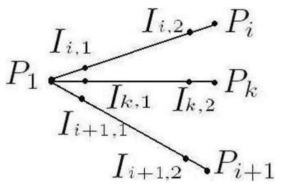

Fig. 3. A new pixel P_k is added between P_i and P_{i+1}.

the strokes and the line $P_1'P_k'$). Consequently, the following equation holds:

$$cross\_ratio(P_1, I_1, I_2, P_k) = cross\_ratio(P_1', I_1', I_2', P_k') \quad (2)$$

For simplicity, we rewrite the notation as $CR(P_1, P_k) = cross\_ratio(P_1, I_1, I_2, P_k)$. When there are more than two intersections between two points, such as P_1 and P_n shown in Fig. 1, only the first two intersections (near P_1) are used. The intuition is that, symbols have very simple structures, and the shapes comprising of the first two intersections already give enough information to differentiate them. Some information of the inner structure of a symbol is lost using this method. However, even if the shape comprising of the first two intersections is not distinctive enough, in theory we still can extend our method to employ other intersections in the same manner. Note that intersections at the convex hull itself are ignored in our implementation of the method, to prevent noise from being introduced by a jagged outer contour. If the number of intersections is 0 or 1 and thus no cross ratio value can be computed, the pseudo-cross ratio value is assigned as -1 and 0 respectively. These two values are chosen because cross ratio values range from 1 to ∞. This assignment is to guarantee that both pseudo-cross ratios are distinct from a real one. A cross ratio spectrum (CRS) is a sequence of cross ratios. The sequence exhibits a wavelet like form when plotted, and thus we call it "cross ratio spectrum". Suppose the sample point sequence of the convex hull of H is $\{P_s, s = [1 : S]\}$, where P_2 is the anti-clock-wise neighbor pixels of P_1, etc. The CRS of a pixel P_i is defined as $CRS(P_i) = \{CR(P_i, P_{i+1}), ..., CR(P_i, P_n), CR(P_i, P_1), ..., CR(P_i, P_{i-1})\}$. Examples of CRSs are shown in Fig. 2.

B. Modeling the Perspective Effect in a Cross Ratio Spectrum

It is easy to know that the spectrum is intrinsically translation and rotation invariant, given the starting point. An example of character 'H' is shown in Fig. $2(a)$, and its perspective variant is shown in Fig. $2(b)$. The CRSs of mapping pixels P_1 and P_1' are also shown in Fig. 2(a) and (b) respectively, where the x-axis is the pixel index and the y-axis is the cross ratio value. The pixels on the contour accounting for peaks shown in the spectrum curves (labeled with Greek characters) are shown. Three observations are made:

- Both spectra are visually similar to each other, but spectrum b has a certain fluctuation on the x-axis.
- The value of peaks β and β' are quite different.
- There is a abnormal peak between α' and β'.

The following explains how the first observation happens. Suppose there are two neighboring pixels $P_i \in P$ and $P_{i+1} \in P$, where P is a fronto-parallel character, as shown in Fig. 3. $I_{i,1}$, $I_{i,2}$, $I_{i+1,1}$, and $I_{i+1,2}$ are the intersections between pixel $P_1 \in P$ and the other two pixels . It is assumed that P_1, P_i and P_{i+1} are on the smooth part of the convex hull (not at the corner), where the stroke width does not change or changes slowly in the neighborhood. Hence $I_{i,1}$ and $I_{i+1,1}$, $I_{i,2}$ and $I_{i+1,2}$ are near to each other, thus $CR(P_1, P_i) \approx CR(P_1, P_{i+1})$. After perspective projection, the segment, to which P_i and P_{i+1} belong, is elongated, new pixels are added between them. Suppose only one pixel P_k is added between P_i and P_{i+1} at the beginning. Similarly, we get $CR(P_1, P_k) \approx CR(P_1, P_i)$. More pixels could be added in the same manner. In short, the cross ratios of those newly added pixels can be approximated by that of the original pixels. Because the number of pixels on the smooth part always dominates, pixels at corners are statistically unimportant. In a nutshell, under perspective transformation, some parts of a character expand, while some parts shrink, which leads to the increase or decrease of the number of pixels on certain parts of the convex hull. As a result, some segments of the spectrum curve are elongated, while others are shortened. Therefore, the perspective deformation in an image can be modeled as an uneven stretching deformation in a spectrum.

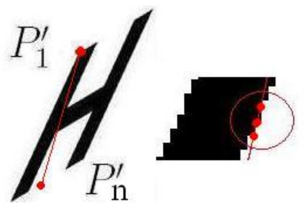

Fig. 4. False intersections on a jagged inner contour.

The following explains how the second and third observations happen. Following Equation (1), suppose $s_1 = P_1P_2$, $s_2 = P_2P_3$, $s_3 = P_3P_4$, thus

$$cross\_ratio(P_1, P_2, P_3, P_4) = \frac{\frac{s_1}{s_2} + 1}{\frac{s_1}{s_2+s_3} + 1} \qquad (3)$$

which is an increasing function of variable s_2. s_2 becomes quite short when the line from P_1 passes through a corner, such as at location β and β', leading to large cross ratio values. β and β' are different due to quantization errors. The reason accounting for abnormal impulses in Fig. 2(b) is jagged inner contour. As shown in Fig. 4, there should be only one intersection between P'_1 and P'_n, but because of the jagged inner contour, false intersections are detected, leading to a very short s_2. A cost function (Equation (5)) is chosen to minimize the impact of this kind of noise.

C. Comparing Cross Ratio Spectra

In Fig. 2, spectrum b is an uneven stretching version of spectrum a. Hence, we use Dynamic Time Warping (DTW) to compare the similarity between spectra a and b. DTW is widely used in speech recognition to eliminate the time-axis fluctuation between the given word and a template [20]. In the following sections, Q refers to an unknown character with a convex hull $Q_i, i = \{1 : M\}$, and T refers to a template character with a convex hull $T_j, j = \{1 : N\}$. The notation of $CRS(Q_i)$ is rewritten as $CRS(Q_i) = \{q_u, u = 1 : M - 1\}$ for simplicity. Similarly, $CRS(T_j) = \{t_v, v = 1 : N - 1\}$. The comparison between two sample points Q_i and T_j is formulated as:

$$DTW(u,v) = min \begin{cases} DTW(u-1, v-1) + c(u,v) \\ DTW(u-1, v) + c(u,v) \\ DTW(u, v-1) + c(u,v) \end{cases} \qquad (4)$$

$$c(u,v) = \frac{abs(log(CR(Q_i, Q_u)) - log(CR(T_j, T_v))}{log(CR(Q_i, Q_u)) + log(CR(T_j, T_v))} \qquad (5)$$

It is observed that large cross ratios are unstable. Also, we found that most cross ratios of symbols are within the range of $(1, 2]$. The $log(CR(.,.))$ representation is used in the cost function $c(.,.)$, in order to reduce the weight of unstable cross ratios and to differentiate common cross ratios within the range of $(1, 2]$ better. If $CR(.,.)$ is -1 or 0, $log(CR(.,.))$ is assigned as -1 and -0.5 respectively. The cost function $c(.,.)$ is chosen to minimize the effect of noise, and to maximize the penalty when a pseudo-cross-ratio is mis-aligned with a real one. The distance between points Q_i and T_j is given by the last item:

$$DTW\_dist(Q_i, T_j) = DTW(M-1, N-1) \qquad (6)$$

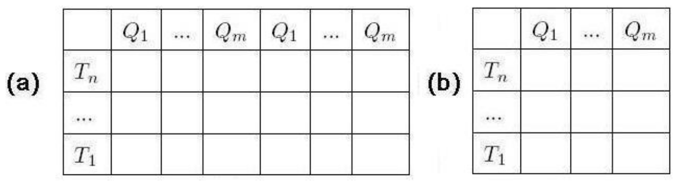

Fig. 5. (a)DTW distance table. (b)Searching in a sub-table.

III. Planar Symbol Recognition

Given $Q = \{Q_1, Q_2, ..., Q_M\}$ and $T = \{T_1, T_2, ..., T_N\}$, the distance between Q and T can be formulated as:

$$dist(Q,T) = \arg\min \sum_{w\_global} DTW\_dist(Q_i, T_j) \qquad (7)$$

where w_global is the global warping path between (Q_1, T_1) and (Q_M, T_N), as well as the correspondence between Q and T. Our strategy to solve this equation is as follows. An arbitrary sample point of T is chosen as the starting point T_1, and T_1 is aligned with each Q_i as the boundary condition. Particularly, DTW comparisons are conducted between each pair of Q_i and T_j, and a DTW-distance-table is constructed in the manner given by Fig. 5(a). Cells in the table denote the distances of corresponding pixel pairs. Each time, a DTW is applied to a sub-table comprising of column $\{\hbar, \hbar+1, ..., \hbar+M-1\}$ of the table, to align T_1 with $Q_\hbar$ and T_N with $Q_{\hbar+M-1}$ as the boundary condition. The comparison is formulated as follows:

$$DTW(i,j) = min \begin{cases} DTW(i-1, j-1) + c(i,j) \\ DTW(i-1, j) + c(i,j) \\ DTW(i, j-1) + c(i,j) \end{cases} \qquad (8)$$

$$c(i,j) = DTW\_dist\_table(\hbar + i - 1, j) \qquad (9)$$

where $i = 1 : M$ and $j = 1 : N$. Fig. 5(b) illustrates a sub-table when $\hbar = 1$. A candidate distance between Q and T is given by $DTW(M, N)$. M DTW comparisons are conducted. Among M candidate distances, the smallest one gives the desirable global distance.

In all experiments in this paper, we take a nearest neighbor recognition strategy: a query is compared with all templates, and the template which has the smallest distance with the query gives the identity of the query.

IV. Synthetic Image Testing

In this section, the ability of handling perspective deformation of the proposed method will be illustrated with a well defined synthetic image set. Scale Invariant Feature Transforms (SIFT) with Harris-Affine detector [1], Shape Context [2], MPEG7 contour shape space descriptor (CSS)[3] and a widely used commercial OCR called Scansoft OmniPage Pro 14.0 (OCR) are employed as comparative methods.

Shape context is a global descriptor, in which each sample point on the shape contour is represented by the distribution

[1]http://www.robots.ox.ac.uk/̃vgg/research/affine/index.html
[2]http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape
[3]http://mpeg7.doc.gold.ac.uk

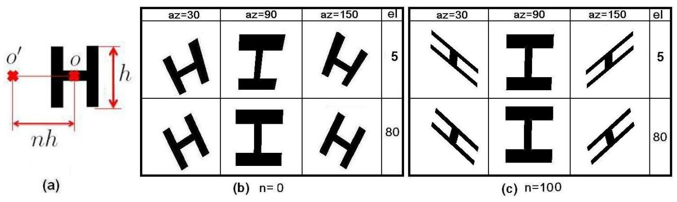

Fig. 6. Samples of synthetic character images.

of the remaining points relative to it, and a point-to-point correspondence between the query and a template is solved by a bipartite graph matching. After that, a Thin Plate Spline model-based transformation is estimated for a better alignment between two shapes. The distance between two shapes is given by a sum of shape context distances. CSS is also a global descriptor, which represents a shape by features of its curvature scale space image, such as the number of peaks, the height of the highest peaks, and the positions of the remaining peaks. The identity of Q is given by the template which has the minimum CSS score with Q. SIFT is an local affine-invariant descriptor which describes a local region around a key point. SIFT descriptor is robust to occlusion, and does not require segmentation. A foreseeable problem of applying SIFT descriptor to symbols is the lack of discriminating power. First, many symbols share similar structural primitives, like concavities of 'N' and 'Z', which are difficult to be distinguished using SIFT. Furthermore, the symmetry of a symbol itself causes ambiguity. False matches may happen among symmetric structural primitives of the same symbol. For example, key points at the left bottom of character 'A' may be matched to points at the right bottom of 'A'. In order to solve the structural ambiguity and maximize the recognition strength of SIFT descriptor, the recognition process is designed as follows. A Harris-Affine detector is used to detect Affine-invariant key points. For each key point of Q, its first 20 nearest neighbors are found in the training set. If the distance is less than a threshold (200 in the experiment), the neighbor is kept, otherwise is thrown away. RANSAC fitting algorithm is then used to further filter false matches. False matches are removed by checking for agreement, between each match and the perspective transformation model generated by RANSAC. The identity of Q is given by the template which has the maximum number of correct matches with Q. Because the Ominipage OCR assumes that the character ready for recognizing should be in a correct orientation, only a slight skew is acceptable. A query image is rotated az degree backward to tentatively remove the skew.

A. Experimental Setup

A template set is trained on synthetic fronto-parallel images of 62 characters, namely 26 uppercase English characters, 26 lowercase English characters, and 10 digits, of Arial font and bold style. 18 testing datasets are generated by Matlab using various perspective parameters. Characters 'l' and 'I' are considered the same in our experiment. The perspective images are generated by setting the target point at a specific point o', and setting the perspective viewing angle as $25°$ (to model a general camera lens), while changing the azimuth (az) and elevation (el) angles gradually. o' is at the same horizontal line as the center of a character, denoted by o, with a distance of $n \times h$, where n is a positive integer and h is the height of the character. An illustration is shown in Fig. 6(a). Generally, the larger the n is, the greater the deformation is. For each testing set, n and el are predefined, and az is set as $\{30°, 90°, 150°, 210°, 270°, 330°\}$ respectively. Therefore, each testing set comprises of $6 \times 62 = 372$ characters. Examples of the character 'H' with different perspective parameters are shown in Fig. 6(b) and (c).

(a) our method

el=	5°	10°	30°	50°	70°	90°
$n = 0$	96.77	97.31	97.04	100	100	100
$n = 50$	96.23	97.04	97.04	100	100	100
$n = 100$	96.23	97.04	97.04	97.84	97.84	100

(b) SIFT

el=	5°	10°	30°	50°	70°	90°
$n = 0$	39.24	45.16	55.91	59.67	67.74	81.18
$n = 50$	11.02	13.70	16.12	17.47	18.81	75.26
$n = 100$	11.55	10.21	11.02	9.94	8.60	65.05

(c) Shape Context

el=	5°	10°	30°	50°	70°	90°
$n = 0$	64.24	67.47	68.27	71.50	73.92	96.23
$n = 50$	15.86	16.93	20.43	18.01	54.03	66.39
$n = 100$	15.32	13.44	15.05	14.51	50.00	68.27

(d) OCR

el=	5°	10°	30°	50°	70°	90°
$n = 0$	92.47	92.47	97.84	97.84	100	100
$n = 50$	0	0	2.68	3.22	3.49	98.65
$n = 100$	0	0	0	0	2.41	98.11

TABLE I
RECOGNITION ACCURACY OF SYNTHETIC IMAGES.

B. Experiment Results

Tables I(a), (a), (c), and (d) show the recognition accuracy using our method, SIFT, Shape Context, and OCR methods respectively, where accuracy is the number of correctly recognized characters over the number of total query characters. The accuracy in each cell is based on a testing set comprising of 372 characters generated with corresponding perspective parameters. The result of CSS is not tabulated, simply because it cannot distinguish simple symbols. CSS descriptor gave the best score to about 10 to 20 templates in each run. However it still showed a certain resistance to moderate perspective deformation.

It is easy to see that when characters are deformed by perspective projection, our method has a better recognition accuracy than other methods. Generally, when n increases or el decreases, the deformation becomes more severe. Table I(a) shows that the performance of our method degraded only a little with increasing deformation. Particularly, the

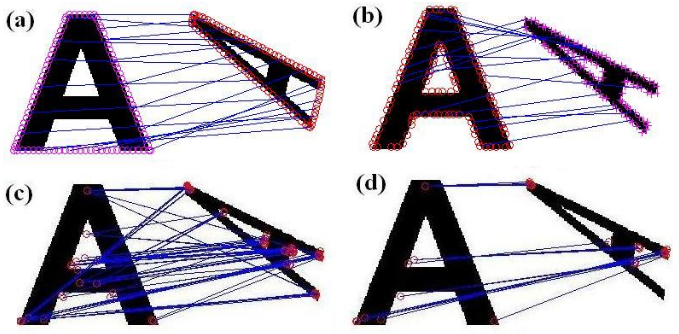

Fig. 7. Pixel level correspondence of a template and a query generated by (a)our method, (b)Shape context, (c)SIFT, (d)SIFT with RANSAC.

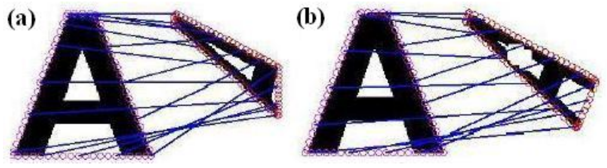

Fig. 8. Pixel level correspondence of a template and impaired queries.

character pair 'O' and 'D' are responsible for most errors. It is worth noting that the proposed method gave small distances to visually similar characters like 'W' and 'w' but made few mistakes in differentiating them. For the performance of SIFT descriptor shown in Table I(b), when the perspective deformation is small, such as $n = 0$ or $el = 90°$, errors are mainly caused by the structural similarity of characters. However, when the deformation is more severe, the descriptor is not resistant to the deformation any longer. An illustration can also be found in Fig. 7(c) and (d). Table I(c) shows that Shape Context is not that robust to severe perspective deformation too. One possible reason is that both SIFT and Shape Context descriptors are statistics based, but the expansion/shrinking effect in perspective deformation obviously will affect the statistics. Table I(d) shows the recognition result gotten by OCR. With necessary preprocessing, OCR achieves an accuracy as high as above 92%, when the deformation is moderate. In particular, 'O' is mis-recognized as '0' for almost all testing datasets with perspective deformation. However, when the deformation becomes more severe, the performance of OCR drops rapidly. One reason for the rapid degradation is that rotating az degree backward may not turn the character into a right position. In this case, only a few characters can be recognized.

Fig. 7 shows the pixel level correspondence, between a query ($az = 30°, el = 5°, n = 100$) and a template, achieved by our method, Shape Context and SIFT respectively. The severe perspective deformation as well as structural similarity of characters fail both Shape Context and SIFT method. Thanks to the flexibility of DTW comparison, our method is tolerant to image defects to a large extent. Fig. 8(a) and (b) illustrate the results of finding correspondence using our method when the exterior contour and interior contour of a character are impaired respectively. The alignment can be almost correctly found.

To compare a query with one template, each of which has 100 sample points, it takes 2.62 seconds for our method implemented in Java, on a PC configured with Pentium 4 CPU 3GHz, 0.99GB of RAM. We also implement the method in Matlab to compare it with other methods. It takes 10.23 seconds for our method(Matlab, 100 sample points), 6.86 seconds for Shape Context(Matlab, 100 sample points), 6.61 seconds for Harris&SIFT&RANSAC(C++/Linux), 0.06 seconds for Contour Shape Space (C++) methods, and 0.26 seconds for OCR.

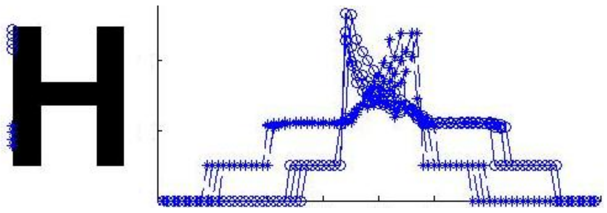

Fig. 9. Neighboring points having similar spectra.

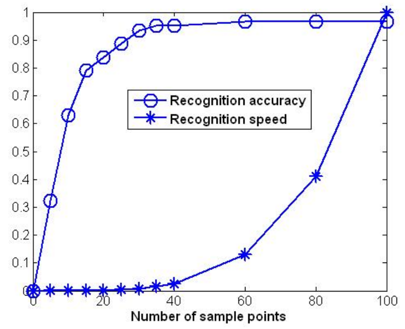

Fig. 10. The recognition accuracy and speed with different number of sample points.

V. Discussion on Speed

The main drawback of the proposed method is the speed. A query Q has to be compared against a whole set of templates. Also, when Q is compared with a template T, comparisons between each pair of T_i and Q_j are needed. Constructing the DTW-distance-table needs $M \times N$ DTW comparisons. Moreover, searching the optimal warping path in the table takes M DTW comparisons. The time complexity of a single DTW comparison is $O(M \times N)$. Therefore, the time complexity for comparing Q and T is $O(M^2 \times N^2)$. An important observation is that many neighboring points have similar spectra. An example is shown in Fig. 9. Two groups of neighboring points are labeled with $*$ or $\circ$ marker respectively, and spectra of points are labeled with the same markers. Spectra of the same group are similar to each other, by shifting in the x-axis. On the contrary, spectra of different groups are different. This phenomenon indicates two possible solutions to the speed problem, namely, reducing the number of sample points for each character and indexing templates by grouping neighboring points together.

A. *Effect of the Number of Sample Points*

It is easy to see that the essential factor which affects the speed is the number of points sampled on the convex contour of a character. In this subsection, this issue will be discussed. The recognition accuracy and speed of sampling $k = \{5, 10, 15, 20, 25, 30, 35, 40, 60, 80, 100\}$ points on each character (both template and query) are shown in Fig. 10. The speed for different k is shown as a ratio to the speed when $k = 100$(2.62 seconds). The result is based on one synthetic dataset (372 queries with parameters el = 5, n = 100). When k is greater than 60, the recognition accuracy remains very close to 96.23%.

We also run the same experiment for another 10 times, randomly changing the starting point of the sample points, and this has little impact on the final recognition accuracy when the number of points is larger than 40. When the number of sample points is less than 20, the recognition accuracy is not stable as the starting point changes. A possible reason is that the sample points are too sparse to capture the shape of a character. Furthermore, we scale characters in the testing dataset by 2, 4, and 8 times respectively, and use the scaled dataset as input. Because scaling will also affect the distribution of sample points. The results show that the size of query will not affect the recognition accuracy, too. In a nut shell, although the speed is bi-quadratic to the number of sample points N, N is nearly fixed in a character recognition application for all sizes of characters. Further, an appropriate N could be estimated by a training process, given that all templates are available.

B. Indexing Templates

Based on our observation that neighboring points have similar CRSs, we employ a KNN clustering method here to reduce the number of points needed to be indexed. CRSs of all 3720 points extracted from the template set (60 points from each of 62 templates) are generated. Pairwise DTW distances are computed for these points. KNN clustering method is applied on these pairwise distances. 100 clusters are formed. The centroid of a cluster is defined as the CRS which has the minimum sum of distances to the other CRSs in the cluster. The centroid and member CRSs for each cluster are recorded. When a query comes in, it is compared to the centroid of each cluster. The results is used to fill up the DTW-distance-table (Fig. 5), referring to the member list of clusters. By doing this, we are able to reduce the times of DTW comparisons from 3720 to 100 in the step of filling up DTW distance tables, which is the most costly step in the whole recognition process.

We apply the clustering method on a synthetic dataset ($el = 5, n = 100$), and achieve an accuracy at 91.64%, using 0.02 seconds to compare a query and a template on average, compared to 2.62 seconds of the original method. We also note that a few DTW speed up methods based on lower bound estimation [7] have been proposed and widely employed in time series database management. We will explore to integrate this technique and further speed up the recognition process in future.

C. Coarse-to-Fine Matching Scheme

Although indexing templates by grouping similar CRSs will greatly reduce the recognition speed, about 5% accuracy is lost consequently, mainly due to some similar characters. Therefore, we here employ a two-level coarse-to-fine matching scheme to further enhance the recognition accuracy, while keeping the speed fast. Firstly, n nearest templates of Q are identified by the indexing algorithm stated in the previous subsection, and then these n templates are re-ranked by the original comparison algorithm.

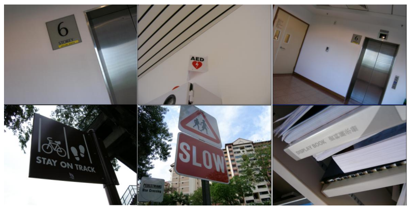

Fig. 11. Samples of sign boards in real scene.

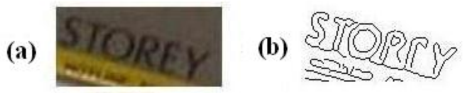

Fig. 12. Difficult testing photos in real scene.

VI. Real-Scene Character Identification

Robust character recognition is a broad research topic. It aims to recognize characters in real scene images and to overcome various difficulties encountered in recognition, including uneven illumination, occlusion, blur, highly decorated fonts, as well as perspective deformation. In this paper, we will try to tackle this problem partially, namely to identify characters in real scene, by handling severe perspective deformation, while keeping other difficulties mentioned above at a moderate level. In view of the fact that comparative methods are not discriminating enough for character recognition in the synthetic experiment, no comparative methods are employed in this experiment.

In this experiment, 100 sign boards are chosen. For each signboard, 4 photos are taken from different angles and distances, leading to 400 photos in total. Examples of these photos are shown in Fig. 11. These photos are divided into training and testing datasets. Training dataset has 1 photo of each signboard, which are clear and nearly fronto parallel. Testing dataset has the other 3. Words are extracted by the method proposed by Chen [3]. In order to avoid errors introduced by the extraction algorithm, non-character elements (here a character means either an English character or a digit) are manually eliminated in both training and testing datasets. Then the edge of a character is extracted by Canny algorithm. In order to remove undesirable edges caused by shadows or dusts on signboards, edges with a length shorter than $e = 0.02 \times \sqrt{s}$ are removed, where s is the area of the character bounding box. We have gotten more than 1000 training characters in several fonts. To remove duplicate characters with identical fonts, a template set is trained from the training set as follows. The template set Γ is initialized as $\Gamma = \emptyset$. Characters are chosen from the training set in an arbitrary order. If a character cannot be recognized with Γ correctly, it is added to Γ, otherwise thrown away. After running this for all characters, a template set, comprising of

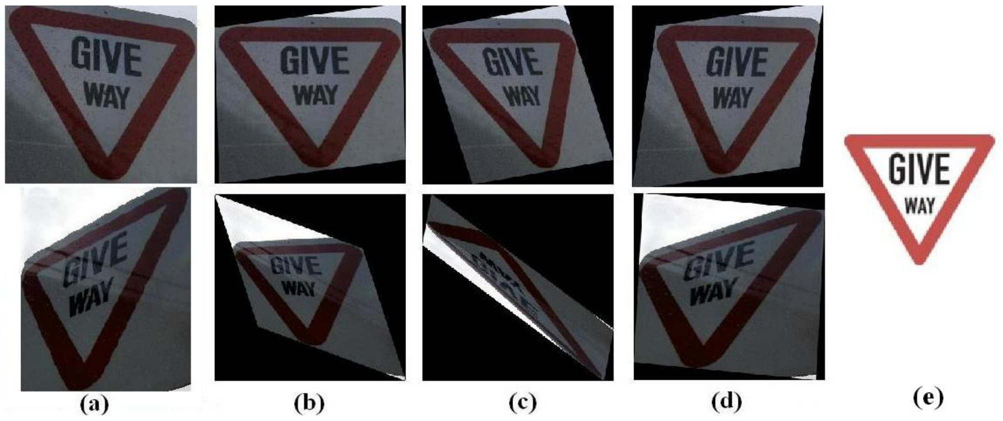

Fig. 14. Rectify photos by the correspondence given by different methods, rectified images are scaled for better viewing purpose. (a) real-scene symbols (b) by our method (c)by SIFT (d)by Shape Context (e) template.

Fig. 13. Samples of traffic sign boards in real scene.

174 characters, is ready for use. The clustering based indexing is applied on the template set. In recognition, after identifying the first $n = 5$ candidates, the full comparison algorithm is used on these candidates.

The testing set is further divided into three sub-sets. Set I has 923 characters, which can fit into a bounding box of 50×50 pixels; Set II has 1296 characters smaller than 100×100 pixels; Set III has the remaining 1026 characters. The result shows that the proposed method achieves an recognition accuracy at 69.55% in the testing Set I, 90.35% for Set II, and 93.37% for Set III. The recognition performance degrades when the character size gets smaller. Because when the character is small, it is very likely that the edge detected will be broken or connected to edges of background objects. For example, when a signboard is far away and there happens to be an object near it, the edge detector may not be able to generate a correct edge of the characters on the signboard, such as shown in Fig. 12(b). With the coarse to fine matching scheme (60 sampled points, 150 indexing clusters, 5 nearest neighbors), it takes 7.02 seconds to process one query over 174 templates on average.

VII. Real Scene Compound Symbol Recognition

Traffic signs are selected in our experiments representatives of compound symbol comprising of several components, for the ease of symbol detection and the availability of a comprehensive template set. A subset of a standard traffic sign database [4](45 signs with red or blue frames) is employed as the template set. Both SIFT and Shape Context, with the same setting in Section IV, are employed for comparison, considering that road signs are more distinctive from each other than characters.

For a compound symbol with several components, a recognition strategy is to disassemble the symbol into components and then recognizing them separately. However, this strategy is not employed here, because the traffic sign segmentation algorithms are already available. We do recommend using an extra segmentation step for multiple-component symbols in order to increase the distinctiveness of symbols. In particular, although the distance is not directly related to the number of components in a symbol, more components often lead to greater curvature in a symbol, and thus traffic signs often have greater average distance as well as greater distance range among each other than characters.

3 photos for each of 100 traffic boards are taken, and examples are shown in Fig. 13. Many of them have elevation angles smaller than 20°, leading to severe perspective distortion of the traffic boards. In the experiment, we employ a simple yet effective color thresholding method proposed in [5] to detect signs with red and blue frames, with hardcoded color boundaries. Desirable traffic signs, 448 in total, are extracted, because some photos have more than one traffic signs. Then the edge is extracted by Canny algorithm, too. Edges with a length shorter than $e = 0.01 \times \sqrt{s}$ are removed, where s is the area of the bounding box. The parameters used in this experiment are 80 sampled points, 100 indexing clusters, and 3 nearest neighbors, based on a preliminary experiment on synthetic traffic symbols.

The testing dataset comprising of 300 photos is divided into two sets: Set I comprising of 256 signs whose sizes are within a 80×80 pixel bounding box, while Set II comprises of the remaining 192 signs. Table II shows the recognition results for our method, SIFT, and Shape Context respectively. The recognition performance of our method is slightly better than character recognition. Two reasons may account for this. First, traffic signs are more distinctive from each other. Second, traffic signs used in the experiment have larger size and are more distinct from the white background, which is good for edge detection. Both SIFT and Shape Context methods give much better recognition results in this experiment than in

[4]http://en.wikipedia.org/wiki/Road_signs_in_Singapore

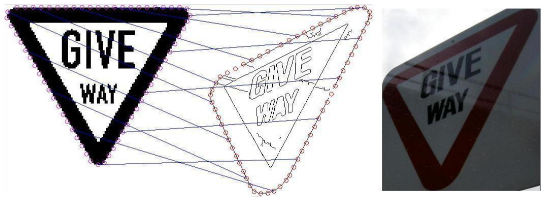

Fig. 15. Pixel level correspondence of a template and a deformed query.

	Our method	SIFT	Shape Context
$SetI$	94.79	72.26	87.89
$SetII$	92.18	72.39	81.25

TABLE II
THE RECOGNITION RESULTS OF TRAFFIC SYMBOLS.

the synthetic image experiment. One possible reason is the distinctiveness of traffic symbols. Another possible reason is that the perspective deformation appearing in most photos in this dataset is not as severe as those used in Section IV. In Section IV, characters with different perspective deformations are evenly distributed. However, due to physical constrains in real scene, traffic signs with severe perspective deformation are only a small fraction of the testing data. Fig. 14 shows the results of rectifying two symbols by ours method, SIFT, and Shape Context respectively, using Least Square method to evaluate a transformation model based on correspondences between a real-scene symbol and the template achieved by the three methods. The major noise is caused by edges of shadows and other objects on the traffic board. Fig. 15 illustrates the alignment of a deformed symbol. Short edges produced by the shadow and rain drops remain after the preprocessing. In order to improve the accuracy, a better way to remove this noise should be included. Also, some parts of the contour of the symbol are missing due to low contrast. However, our method is still able to give a correct alignment. The proposed method is also capable of giving a correct alignment between similar symbols. An example is shown in Fig. 16, where the parts beneath the person are different between the two signs. In the experiment, it takes 2.64 seconds to process each query over 45 templates on average.

VIII. CONCLUSION

This paper introduces a new planar symbol recognition method based on a global descriptor named Cross Ratio Spectrum, which is robust to severe perspective deformation. One issue that has not been addressed in this paper is the occlusion condition. It is a common problem for all global descriptors. We will solve this problem in future, by adapting CRS descriptor to a local image patch instead of a whole symbol.

ACKNOWLEDGMENT

This research is supported in part by IDM R&D grant R252-000-325-279. Thanks to Eamonn Keogh for his suggestions on DTW indexing.

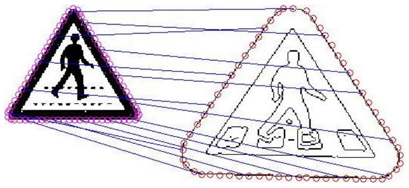

Fig. 16. Pixel level correspondence of two similar, but not identical, symbols.

REFERENCES

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24:509 – 522, 2002.

[2] M. Bober. MPEG-7 visual shape descriptors. *IEEE Trans. on Circuits and Systems for Video Technology*, 11:716 – 719, 2001.

[3] X. Chen and A. L. Yuille. Detecting and reading text in natural scenes. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*.

[4] P. Clark and M. Mirmehdi. Recognizing text in real scenes. *Int'l J. Document Analysis and Recognition*, 4(4):243–257, 2004.

[5] A. de la Escalera, L. Moreno, M. Salichs, and J. Armingol. Road traffic sign detection and classification. *IEEE Trans. on Industrial Electronics*, 44(6), 1997.

[6] D. Jelinek and C. J. Taylor. Reconstruction of linearly parameterized models from single images with a camera of unknown focal length. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23:767–773, 2001.

[7] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proc. of the 8th Int'l Conf. on Knowledge Discovery and Data Mining*, pages 102 – 111, 2004.

[8] L. Li and C. L. Tan. Character recognition under severe perspective distortion. In *Proc. of the 19th Int'l Conf. on Pattern Recognition*, 2008.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int'l J. Computer Vision*, 2(60):91–110, 2004.

[10] S. Lu, B. M. Chen, and C. C. Ko. Perspective rectification of document images using fuzzy set and morphological operations. *Image and Vision Computing*, 23(5):541–553, 2005.

[11] S. Lu and C. L. Tan. Camera text recognition based on perspective invariants. In *Proc. of the 18th Int'l Conf. on Pattern Recognition*, volume 2, pages 1042–1045, 2006.

[12] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(10), 2005.

[13] J. L. Mundy and A. P. Zisserman. *Geometric invariance in computer vision*. MIT Press, 1992.

[14] G. K. Myers, R. C. Bolles, Q. T. Luong, and J. A. Herson. Rectification and recognition of text in 3-D scenes. *Int'l J. Document Analysis and Recognition*, 7(2-3):147–158, 2005.

[15] C. Orrite and J. E. Herrero. Shape matching of partially occluded curves invariant under projective transformation. *Computer Vision and Image Understanding*, 93(1):34–64, 2004.

[16] M. Pilu. Extraction of illusory linear clues in perspectively skewed documents. In *Proc. of IEEE on Computer Vision and Pattern Recognition*, volume 1, pages 363–368, 2001.

[17] C. A. Rothwell, A. Zisserman, D. A. Forsyth, and J. L. Mundy. Planar object recognition using projective shape representation. *Int'l J. Computer Vision*, 16(1):57–99, 1995.

[18] T. Suk and J. Flusser. Vertex-based features for recognition of projectively deformed polygons. *Pattern Recognition*, 29(3):361–367, 1996.

[19] T. Suk and J. Flusser. Projective moment invariants. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(10), 2004.

[20] K. Wang and T. Gasser. Alignment of curves by dynamic time warping. *The Annals of Statistics*, 25(3):1251–1276, 1997.

[21] D. Xu and H. Li. 3-D projective moment invariants. *J. Information and Computational Science*, 4(1), 2007.

[22] T. Yamaguchi, M. Maruyama, H. Miyao, and Y. Nakano. Digit recognition in a natural scene with skew and slant normalization. *Int'l J. Document Analysis and Recognition*, 7(2-3):168–177, 2005.