# Symbol Recognition Using Bipartite Transformation Distance and Angular Distribution Alignment

Feng Min, Wan Zhang, and Liu Wenyin

Department of Computer Science,
City University of Hong Kong, China
{emmwcity, wanzhang, csliuwy}@cityu.edu.hk

**Abstract.** In this paper, we present an integrated system for symbol recognition. The whole recognition procedure consists of image compression, denoising and recognition. We present a pixel-based method to calculate similarity between two symbols using the *bipartite transformation distance* after they are aligned by their angular distributions. The proposed method can overcome some shortcomings of other pixel-level methods. We also propose a new denoising technique in our system to improve the recognition precision and efficiency. Evaluation results on test sets provided by the 2nd IAPR contest on symbol recognition show good performance of the system in recognizing symbols with degradation and affine transformation.

## 1 Introduction

Symbol recognition is an important problem in many application fields, such as recognition of engineering drawings [4], circuit diagrams [5], and handwritten characters [8]. As there are many factors which can influence the performance of a recognition approach, such as affine transformations, distortion and degradation, it is necessary to provide a universal system to preprocess and recognize symbols under all kinds of circumstance.

In this paper we present a pixel-based approach to symbol recognition, which is a brute-force method. The *bipartite transformation distance* is proposed in this paper to calculate similarity in our method. However, it is not invariant to rotation transformation. To recognize symbols with rotation transformation, we compute their angular distributions and align them by their orientations.

We also describe our symbol recognition system used in the GREC 2005 contest. Our system utilizes certain techniques to predigest the input symbol to achieve higher recognition accuracy, including image compression and edge extraction. Various kinds of noise may exist in the input symbols, which severely affect similarity computing. Hence, we also propose a denoising technique to remove noise from the symbols. Experiments show that the integrated system can recognize the symbols effectively and efficiently.

The structure of this paper is described as follows. Related work is reviewed in Section 2. Section 3 presents the proposed methodology. In Section 4, we

W. Liu and J. Lladós (Eds.): GREC 2005, LNCS 3926, pp. 398–407, 2006.

illustrate the steps in our symbol recognition system. Evaluation results are briefly presented in Section 5. Finally, we draw conclusions and discuss future work in Section 6.

## 2 Related Work

One of the most popular classes of recognition methods is pixel-based method. The representative approaches are Fourier descriptors [9], moment invariants [3], ring projection [8] and shape contexts [1]. These descriptors are invariant to affine transformation but each of them has the shortcomings mentioned in [1], [7]. Fourier descripter only describes the external contour and ignores the internal contour and is difficult to be extracted from a real image. Moment invariants are rotation, scale and translation invariant, however, they are unable to provide a detailed profile about a symbol's structure. Hence, their discrimination power is limited. Shape context is a robust descriptor, but its rotation invariance depends on the tangent at every pixel, which is sensitive to deformation.

Intuitively shape matching can be done by matching the point sets extracted from the images. Let $A$ and $B$ be point sets of sizes $n$ and $m$ extracted from two shapes accordingly. A frequently used dissimilarity measure is Hausdoff distance [6]. It is defined as infimum of the distances of the points in $A$ to $B$ and the points in $B$ to $A$. If the two point sets have the same size, i.e., $m = n$, *minimum weight (distance) matching* can be applied [6]. A *minimum weight matching* minimizes the sum of the weights of the edges of the matching. The difference between our approach and the approaches mentioned in [6] is that the assumption in our method is many to many correspondence among the points and different distance functions are implemented.

## 3 Symbol Similarity

### 3.1 Normalizing Symbol

Assume that a 2D symbol is represented as a binary image. Its grey-scale image, $S(x, y)$, can be discretized into binary values as follows:

$$S(x,y) = \begin{cases} 1 & \text{if (x,y) is foreground pixel of the symbol} \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

We can derive the centroid of the symbol, as denoted by $cen(x_0, y_0)$, which is the geometric center of the symbol:

$$\begin{aligned} x_0 &= \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x * S(x,y)\,\mathrm{d}x\,\mathrm{d}y}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} S(x,y)\,\mathrm{d}x\,\mathrm{d}y} \\ y_0 &= \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} y * S(x,y)\,\mathrm{d}x\,\mathrm{d}y}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} S(x,y)\,\mathrm{d}x\,\mathrm{d}y}, \end{aligned} \tag{2}$$

and subsequently, translate the origin of our reference coordinate to this centroid. Next, we define

$$R_{avg} = \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} r(x,y) * S(x,y)\,\mathrm{d}x\,\mathrm{d}y}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} S(x,y)\,\mathrm{d}x\,\mathrm{d}y}, \tag{3}$$

where $r(x,y)$ is the distance between $(x,y)$ and centroid. In other words, $R_{avg}$ is the average radius of all the points in the symbol. To normalize the symbol, we scale it into fixed size by the following formula

$$S'(x,y) = S(x * R_{avg}, y * R_{avg}), \tag{4}$$

where $S'(x,y)$ represents the normalized symbol. After normalized, the symbols matching is invariant to scaling.

### 3.2 Similarity Between Two Symbols

Generally the similarity between two symbols is inverse to the difference between them. In this paper, the difference between two symbols is referred to as *bipartite transformation distance*, which is something like the dissimilarity function in shape matching [2].

Firstly, we define the distance from point $p(x,y)$ to symbol $S$ as the distance between $p(x,y)$ and the closest point $p_s(x_s,y_s)$ in $S$,

$$dis(p,S) = \min_{S(p_s)\neq 0} distance(p,p_s). \tag{5}$$

Given two symbols $S_1$, $S_2$, the cost of transforming $S_1$ into $S_2$ is defined as follows,

$$cost(S_1,S_2) = \int_R dis(p,S_2)^2 * S_1(p)\,\mathrm{d}p, \tag{6}$$

where R is the range of the coordinate of $S_1$. In other words, the cost is equivalent to the amount of work required to transform one symbol into the other. The transformation approach is to move each point in $S_1$ to $S_2$ through the shortest route, and therefore the cost is equivalent to the sum of the distances between each point pair. We normalize the cost by the area of $S_1$ and derive the *unipartite transformation distance* as follows,

$$unidis(S_1,S_2) = \frac{cost(S_1,S_2)}{\int_R S_1(p)\,\mathrm{d}p}, \tag{7}$$

where $unidis(S_1,S_2)$ is the *unipartite transformation distance* from $S_1$ to $S_2$ and the denominator is the number of foreground pixels in $S_1$.

*Unipartite transformation distance* cannot be considered directly as the difference between symbols because of partial matching. For example, if $S_1$ is similar to part of $S_2$, the *unipartite transformation distance* will be very small, but in fact, the two symbols are probably very different and the *reverse transformation distance* is quite large.

Hence we adopt *bipartite transformation distance* to estimate the difference between $S_1$ and $S_2$ instead, which is defined as,

$$bidis(S_1, S_2) = \frac{cost(S_1, S_2) + cost(S_2, S_1)}{\int_R S_1(p)\,\mathrm{d}p + \int_R S_2(p)\,\mathrm{d}p}. \tag{8}$$

The similarity between $S_1$ and $S_2$ is defined as follows,

$$sim(S_1, S_2) = \frac{1}{1 + bidis(S_1, S_2)}. \tag{9}$$

$Sim(S_1, S_2)$ is between 0 and 1. When two symbols are the same, the *bipartite transformation distance* is 0 and the similarity is 1.

### 3.3 Symbols with Rotation

If symbols are rotated, we cannot use the above method directly because it is not rotation-invariant. A simple way is to try rotating one of the symbols several times and find the maximum similarity between them. Since calculating similarity after each rotation costs too much, it is necessary to design a low-cost measure to normalize the orientations of the symbols. We can also apply the gradient-descent algorithm in order to improve the recognition accuracy.

We transform the original reference Cartesian coordinate system into the polar coordinate system based on the following relations:

$$\begin{cases} x = \gamma \cos\theta \\ y = \gamma \sin\theta \end{cases}. \tag{10}$$

Hence,

$$S(x, y) = S(\gamma \cos\theta, \gamma \sin\theta),$$

where $\gamma \in [0, \infty)$, $\theta \in [0, 2\pi)$.

For any fixed $\theta \in [0, 2\pi)$, we then compute the following:

$$g(\theta) = \frac{\int_0^\infty r * S(\gamma \cos\theta, \gamma \sin\theta)\,\mathrm{d}\gamma}{\int_0^\infty S(\gamma \cos\theta, \gamma \sin\theta)\,\mathrm{d}\gamma}. \tag{11}$$

The resulting $g(\theta)$, which can be viewed as a 1-D symbol that is directly transformed from the original 2-D symbol, shows the distribution of the foregound pixels in symbol along angle $\theta$. We assume that $g(\theta)$ is a periodic function and its period is $2\pi$. If the symbol is rotated by angle $\varphi$, its distribution function can be easily represented as $g(\theta - \varphi)$. We do not need to recompute the function $g(\theta)$ after each rotation and therefore this feature can be utilized to judge whether two symbols are in the same orientation.

Given two symbols and their distribution functions, we define the difference between the two distributions:

$$diff\,(\varphi) = \int_0^{2\pi} [g_1(\theta - \varphi) - g_2(\theta)]^2\,\mathrm{d}\theta, \tag{12}$$

where $\varphi$ is the angle by which the symbol $S_1$ is rotated. Our idea is that one of the two symbols must be rotated to make their difference minimum. The rotated angle is denoted as $\varphi_{min}$. Our algorithm to find $\varphi_{min}$ is shown as follows:

Step 1. Input $S_1$ and $S_2$, initialize $run\_length = 2 * \pi/60$, $\varphi = 0.0$, $diff_{min} = \infty$, $\varphi_{min} = 0.0$;
Step 2. Compute the distribution $g_1$ and $g_2$ using Eq(11). In practice, $g_1$ and $g_2$ are represented by discrete arrays;
Step 3. Compute $diff(\varphi)$ using Eq(12);
Step 4. If $diff(\varphi) \geq diff_{min}$, then go to step 7;
Step 5. $diff_{min} = diff(\varphi)$, $\varphi_{min} = \varphi$;
Step 6. Utilize the gradient descent algorithm to minimize the difference;
Step 7. $\varphi = \varphi + run\_length$;
Step 8. If $\varphi < 2 * \pi$, then go to step 3;
Step 9. Return $\varphi_{min}$.

## 4 Symbol Recognition System

To demonstrate the effectiveness of our approach, we implement a symbol recognition system. Given a test symbol, our system can find the best matching one from a set of models.

The input to the system is a set of test symbols and a set of model symbols, which are represented in binary images. We then apply some preprocessing techniques to them and compute similarity between any pair of them. Finally, we output the best matched model for each test symbol as the result. The steps in our system are outlined as follows:

Step 1. Compress the input image. After compressed, each pixel indicates the density of the foreground pixels in the original image.
Step 2. Utilize a newly proposed approach to remove noise from the image.
Step 3. Compute the similarity between any pair of test symbols and models.
Step 4. Output the best matched model for each test symbol.

Next, we present the steps in more details.

### 4.1 Image Compression

Because our method to compute similarity is based on pixels, the number of pixels in a symbol gives a great impact on its time complexity. Therefore, the aim of preprocessing is to decrease the number of pixels while the information in the image loses slightly.

Assume that the input image is a $512 * 512$ 1-bit bitmap, as shown in Fig.1. It is represented as $S(x, y)$, defined in Eq. 1. We compress the original image into a grey-scale image by the following formula:

$$D(x', y') = \frac{\sum_{x=\eta*x'}^{\eta*x'+\eta-1} \sum_{y=\eta*y'}^{\eta*y'+\eta-1} S(x, y)}{\eta^2}, \tag{13}$$

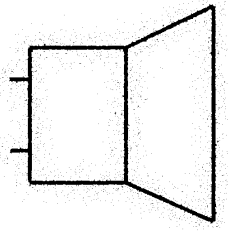

**Fig. 1.** A sample of $512 * 512$ input image

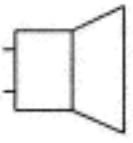

**Fig. 2.** $52 * 52$ compressed image

where $D(x', y')$ is the compressed image shown in Fig.2, $x' \in [0, \lceil \frac{512}{\eta} \rceil]$, $y' \in [0, \lceil \frac{512}{\eta} \rceil]$, and $\eta^2$ is the scaling factor. The value of $D(x, y)$ indicates the density of black pixels in the original image. We can see that the size of the image becomes very small after compression, while little information is lost. Hence, we can compute the similarity based on the compressed images to accelerate the whole process. Moreover, it is also a pre-requisite of our denoising approach.

### 4.2 Denoising

The input images, which contain the symbols, may have various kinds of noise. Noise may severely affect similarity computing, even making the result rather different from the correct answer. Hence, before computing the similarity, we must remove noise from the images.

In our denoising approach, $unidis(S_1, S_2)$ defined in Eq. 7 is the core function. Given a compressed image $D(x, y)$, we denoise it in the following steps:

Step 1. Build another image $S_b$ of the same size with all pixels being black.
Step 2. Compute $unidis(S_b, D)$.
Step 3. If $unidis(S_b, D)$ is smaller than threshold $\omega$, the denoising process ends. In our system, the experimental value of $\omega$ is set to 0.023.
Step 4. Remove the minimal D-valued pixel(s) in the compressed image $D$, turn to Step 2.

We assume that the density of noisy pixels is always lower than that of the foreground pixels in the symbols. Further since the $D(x, y)$ indicates the density of foreground pixels in the original image, removing the minimal D-valued pixels in the compressed image is equal to denoising. When the image is full of noise, the $unidis(S_b, D)$ is rather high. We go on denoising the image until $unidis(S_b, D)$ is lower than the threshold value $\omega$, which is the result of training with a large set of test data. Fig.3 and Fig.4 show two denoising samples.

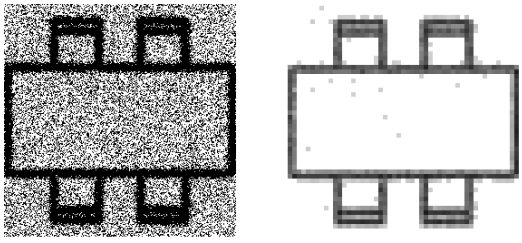


**Fig. 3.** A symbol full of global noise and its denoised image

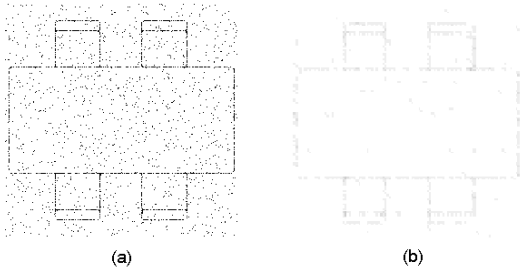


**Fig. 4.** A much degraded symbol with loss of connection and its denoised image

### 4.3 Computing Similarity

Finally we utilize the approach proposed in Section 3 to compute the similarity $sim(D_1, D_2)$ between each pair of preprocessed test symbol and model. After that, the system outputs the best matched model for each test symbol.

## 5 Evaluation

Tab.1 shows the result which our system obtains in the GREC 2005 contest. Our system achieves the highest overall score while the time it costs is the shortest. The tests are grouped according to the type and degree of degradation applied to the images involved in that test. Six degradation models are used and the test symbols may be rotated/scaled.

Among the six degradation models, our system achieves the highest score except for the second model. The reason is that the parameters in our denoising process are not adapted in such kind of noise. A large amount of noise remains after preprocessing, and therefore the result of similarity computing is rather bad. Our accracy in the sixth degradation model is not high either, while other systems score even worse due to severe degradation.

Regardless of noise, neither rotation transformation nor scaling transformation affects much the recognition rate of our system. However, if both exist, our system's accuracy is affected.

**Table 1.** The recognition accuracy of our system for different test sets used for the GREC 2005 contest

| | mod 1 | mod 2 | mod 3 | mod 4 | mod 5 | mod 6 |
|---|---|---|---|---|---|---|
| without rotation and scaling | **0.997** | 0.700 | **0.997** | **0.983** | **0.980** | **0.867** |
| only with rotation | **0.950** | 0.637 | **0.950** | **0.967** | **0.977** | **0.513** |
| only with scaling | **0.965** | 0.385 | **0.960** | **0.930** | **0.915** | **0.620** |
| with rotation and scaling | **0.870** | 0.635 | **0.885** | **0.895** | **0.830** | **0.335** |

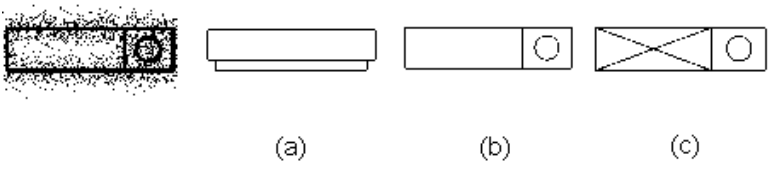


**Fig. 5.** Similarity between test symbol and three model symbols. (a) 0.993341 (b) 0.999193 (c) 0.995259.

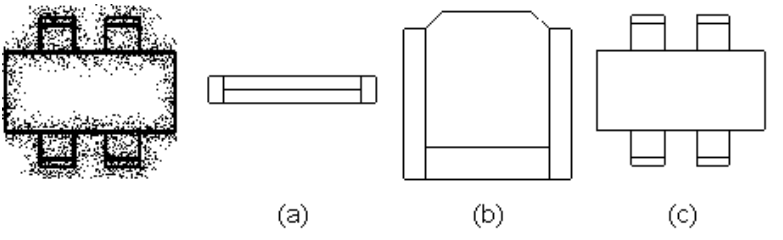


**Fig. 6.** Similarity between test symbol and three model symbols. (a) 0.869658 (b) 0.990119 (c) 0.999074.

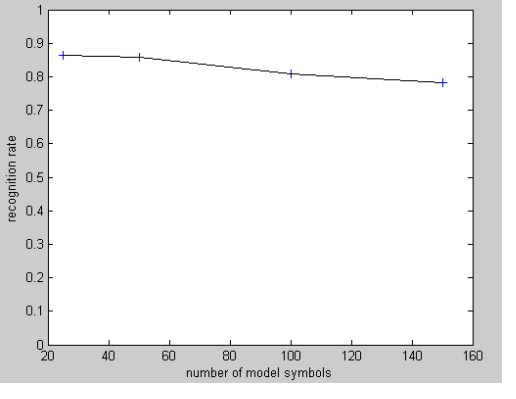


**Fig. 7.** Result with different numbers of model symbols

Fig.5 and Fig.6 are two groups of symbols from the contest. In each group, the leftmost symbol is the test symbol with some noise and the rest three symbols are model symbols. All the three model symbols in the first group are very similar to the test symbol, which may be even confusing to human beings, especially (b) and (c). The similarities given by our system is reasonable, which reflect the real ranking result of the similarities among them. In the second group, the result looks also very good.

Fig.7 shows the curve of average recognition rates over the number of model symbols of a test set. There are 150 different model symbols available in GREC 2005. Each test consists of 4 sets, with 25, 50, 100, 150 symbol models, respectively. As shown above, the recognition rate decreases slightly as the number of symbol models increases. We can conclude that each additional model symbol reduces the discrimination ability of our system by 0.07% averagely.

## 6 Conclusion

We have presented a symbol recognition system, which employs computing bipartite transformation distance, rotating symbols according to the angular distribution, compressing images and a newly-proposed denoising approach. The evaluation in GREC 2005 shows that the combination of these techniques is effective and efficient.

We note that there are still some improvements to be pursued in future work. First, we can utilize the vectorial descriptions of the model symbols. Currently, vectorial descriptions are ignored. If we can retrieve some descriptors from them, the recognition precision may be improved greatly. Second, the angular distribution in our method has some features suitable for other symbol recognition approaches. It is potential and will be studied in the future.

## Acknowledgement

The work described in this paper is fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 1147/04E].

## References

1. S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.
2. P. J. Best and N. D. Mckay. A method for registration of 3d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.
3. M.K. Hu. Visual pattern recognition by moment invariants. *IRE Trans. on Information Theory*, 8:179–187, 1962.
4. Y. Luo and W.Y. Liu. Engineering drawings recognition using a case-based approach. In *Proceedings of Seventh ICDAR*, volume 1, pages 190–194, Edinburgh, UK, 2003.

5. A. Okazaki, S. Tsunekawa, T. Kondo, K. Mori, and E. Kawamoto. An automatic circuit diagram reader with loop-structure-based symbol recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(3):331–341, 1988.
6. R. C. Veltkamp and M. Hagedoorn. State-of-the-art in shape matching. Technical report uu-cs-1999-27, Utrecht University, the Netherlands, 1999.
7. S. Yang. Symbol recognition via statistical integration of pixel-level constraint histograms: a new descriptor. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27:278–281, 2005.
8. P.C. Yuen, G.C. Feng, and Y.Y. Tang. Printed chinese character similarity measurement using ring projection and distance transform. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 12(2):209–221, 1998.
9. C. Zahn and R. Roskies. Fourier descriptors for plane closed curves. *IEEE Trans. Computers*, C-21(3):278–281, 1972.