Rotation-Robust Math Symbol Recognition and Retrieval Using Outer Contours and Image Subsampling

Siyu Zhu[a], Lei Hu[b] and Richard Zanibbi[b]

[a]Center for Imaging Science, Rochester Institute of Technology, Rochester, USA;
[b]Department of Computer Science, Rochester Institute of Technology, Rochester, USA

ABSTRACT

This paper presents an unified recognition and retrieval system for isolated offline printed mathematical symbols for the first time. The system is based on nearest neighbor scheme and uses modified Turning Function and Grid Features to calculate the distance between two symbols based on Sum of Squared Difference. An unwrap process and an alignment process are applied to modify Turning Function to deal with the horizontal and vertical shift caused by the changing of staring point and rotation. This modified Turning Function make our system robust against rotation of the symbol image. The system obtains top-1 recognition rate of 96.90% and 47.27% Area Under Curve (AUC) of precision/recall plot on the InftyCDB-3 dataset. Experiment result shows that the system with modified Turning Function performs significantly better than the system with original Turning Function on the rotated InftyCDB-3 dataset.

Keywords: Machine-Printed Math Symbol Recognition, Content-Based Math Symbol Retrieval

1. INTRODUCTION

Recognition of math expression can be used in a variety of contexts.[1,2] For example, it can help convert math formulas in scientific paper document into their corresponding electronic forms. Recognition of math expression consist of two parts: symbol recognition and structural analysis. Many structural analysis methods assume the symbol class is known and correct before the parsing, making symbol recognition is the basis of structural analysis. Optical character recognition (OCR) for math symbols is difficult[1] because: (1) the number of different symbol classes is large; (2) in the same math expression, math symbols can have different fonts, such as regular, roman, italic, bold and calligraphic; (3) math symbols can have different sizes in the same expression.

Content-Based Image Retrieval (CBIR)[3,4] aims to locate images that are perceptually similar to the query-based on visual content of the images in large databases without the help of annotations. Compared to search engines based on text, CBIR is desirable when people's information need is better expressed by images than by words. Specifically, CBIR is preferred when people are looking for a specific object/scene or a narrow category of objects/scenes that the instances belonging to this category are similar by appearance. The 'content' in CBIR refers to shape, color, texture or any other information which could be extracted from the image.[5] This content can be used as features for retrieval.[3,4] The features used in math symbol retrieval also can be applied to the offline math symbol recognition.

Image-based math symbol retrieval tries to locate symbols that are most visually similar to a query symbol. It can be used to locate math symbols in document image corpora using CBIR. We can use an incremental search or consult the index of the text to find a term. But this cannot be easily done for math symbol because many math symbols are hard to be typed in. It is difficult to construct a text query for math symbol, as people have to remember the name of the math symbol to use the text search. It would be very desirable that people can locate or search an specific mathematical symbol in a large set of documents based on its appearance, and also complete math expression.[6] In recent years, work in mathematical information retrieval has begun, with an emphasis on search within digital libraries for mathematics.[1]

Further author information: (Send correspondence to Siyu Zhu)
Siyu Zhu: E-mail: sxz8564@rit.edu, Telephone: 1 585 475 4536
Lei Hu: E-mail: lei.hu@rit.edu
Richard Zanibbi: E-mail: rlaz@cs.rit.edu

In this paper, we present an unified framework for math symbol recognition and retrieval for the first time based on nearest neighbor analysis. Modified Turning Function (TF) and Grid Features are used to calculate the distance between two symbols based on Sum of Square Distance (SSD). Turning Function[7] is the tangent angle as a function of arclength measured counterclockwise from the starting point. Grid feature is the feature vector of the percentage of black pixels in each cell after we divide each binary symbol image into 5 by 5 rectangular grids. We apply an unwrap process and an alignment to modify Turning Function to make it robust against rotation and changing of starting point. Our system is simple but effective. Compared to existing methods, our system achieves competitive recognition rate and better retrieval performance. The limitation of our system is that it is slow due to the nearest neighbor scheme and large candidate library.

2. RELATED WORK

2.1 OCR for Typeset Math Symbols

A number of approaches have been proposed for offline math symbol recognition in the past two decades. Table 1 shows the summary of these methods.

Table 1: Existing Methods for Classification of Typeset Math Symbols

Paper	**Classifier**	**Feature**	**Dataset**	**(Best) Recognition rate**
Fateman et al.[8]	nearest neighbor	Grid Features, aspect ratio and absolute height	(unavailable)	(unpublished)
Alvaro et al.[9]	nearest neighbor	Euclidean distance between pixel	InftyCDB-1[10]	94.24%
Suzuki et al.[11]	cascading classifier	aspect ratio, crossing features, directions features, peripheral features and Grid Features	about 500 pages of math documents InftyCDB-3	95.18% 96.10%
Malon et al.[12]	cascading classifier[11] + binary SVM	directions histogram of contour and bitmap-like feature	InftyCDB-3	97.70%
Garain et al.[13]	combination of 4 classifiers	stroke features, Grid Features and crossing features	Garain's corpus[17]	93.77%
Alvaro et al.[14]	•HMM •nearest neighbor •weighted nearest neighbor •multi-class SVM	Grid Features distance between pixel distance between pixel (unpublished)	InftyCDB-1[10]	92.7% 96.7% 97.2% 97.4%

Many methods are based on nearest neighbor classification. For offline character recognition, algorithms based on nearest neighbor scheme can get high accuracy but is slow.[18] Fateman et al.[8] proposed a classifier based on nearest neighbor classification in a space of 27 features. For each symbol, its bounding box is divided a 5 by 5 rectangular grid and the percentage of black pixels in each cell will be extracted as a feature. The other two features are aspect ratio and the absolute height in pixels of the bounding box. Euclidean distance is calculated as the dissimilarity between two symbols. For Alvaro et al.,[9] each bounding box was normalized to a fixed size and the Euclidean distance between two images is calculated based on the difference of each pixel. This method obtained a 94.24% symbol recognition rate in InftyCDB-1 database.[10]

Many have used Cascading classifiers. Suzuki et al.[11] employ a three-step coarse-to-fine cascade. The first classifier uses aspect ratio and crossing features. Crossing features are the number of horizontal and vertical switches from black to white or from white to black when we scan through the rows and columns of a math symbol image. Secondly, five candidates will be selected based on a histogram of 36-dimensional directional features. The directional features specify the direction of the line between two consecutive points on the contour. At the last step, three distances based on three sets of features, the above 36-dimensional directional features, 64-dimensional peripheral features and 64-dimensional Grid Features, will be calculated and used for ranking the five candidates. We believe that the peripheral features are the peripheral shape information extracted from along the contour of the symbol. This cascading classifier achieved 95.18% symbol recognition rate on their own dataset. In order to improve the symbol recognition accuracy, Malon et al.[12] adds binary support vector machines (SVM) classifier to the pure Infty engine[11] to distinguish symbols with similar appearance. Based on the confusion matrix produced by the pure Infty engine, a confusion cluster is formed for each symbol class. Assuming each row is the recognition result and each column is the ground truth in the confusion matrix, the set of nonzero entries on each row represents a confusion cluster. The confusing pairs are used to train binary SVM classifiers. Two sets of features: directional histogram of contour (different grid size for different symbol) and bitmap-like feature are used in this classifier. It seems that the bitmap-like feature takes into account each pixel pair's difference and is the same as the feature used by Alvaro et al.[9] This paper uses InftyCDB-3-A for training, and InftyCDB-3-B for testing. Without SVM, the pure Infty engine[11] gets 96.10% accuracy on this testing data set. Using SVM, the recognition rates rises to 97.70%.

Garain et al.[13] proposed a two-level system with four different classifiers. Classifier 1 is used at the first level and employs stroke-based classification technique to recognize frequent characters with simple shapes. It can recognize 83 symbols by looking for the presence of certain strokes. The second level combines three classifiers to recognize the symbols which cannot be recognized by the first level. Classifier 2 uses run-number based feature vectors. The run-number feature vectors are the number of row-wise and column-wise white-to-black transitions and they are similar to the crossing features. Classifier 3 is a nearest neighbor classifier. Its features are the same as the ones used by Fateman et al.[8] Classifier 4 uses Daubechies wavelet transform. Three combination techniques, highest rank, Borda count and logistic regression, have been used to combine those three classifiers to achieve high recognition accuracy. There are 2459 expressions in their own dataset[17] which is extracted from different branches of science. The recognition rates for the combination by the highest rank method,, Borda count and logistic regression are 93.26%, 92.03% and 93.77%. The combination of Borda count performs the worst because it does not consider the individual classifiers' different abilities.

Alvaro et al.[14] presented another offline math symbol recognition method based on Hidden Markov Models (HMM). The image is divided into 25 cells. For each cell, three features, normalized grey level, horizontal gray-level derivative and vertical grey-level derivative, are extracted. The feature extraction is the same as Espana-Boquera et. al.[19] Columns of cells are processed from left to right and a feature vector is built for each column by connecting the features computed in its cells. They compared four techniques, k-Nearest-Neighbor (k-NN), SVN, weighted nearest neighbor (WNN) and HMM. k-NN uses the Euclidean distance between two images taking into account the difference between each pixel. SVM is the multi-class SVM. The WNN technique is an improvement of the classical 1-NN. By using the 1-NN rule with a training set, a discriminative technique is used to learn a weighted distance. Both the prototypes and the features were weighted for each class. Experiments are carried out on InftyCDB-1 data set. The best recognition rates for 1NN, WNN, SVM and HMM are 96.7%, 97.2%, 97.4% and 92.7% correspondingly.

We can find the nearest neighbor classifier and Cascading classifier are commonly used for the classification of typeset math symbols. Because typeset symbols have small variation, the Euclidean distance between two symbols in the feature space can show the difference between them (symbols from the same symbol class will have very small distances while symbols from different symbol classes will have large differences). The number of symbol class is usually large. While single classifier is not sufficient to distinguish all the symbols at one time, cascading classifiers are used to speed up the classification and improve the recognition rate.

Grid features, crossing features and directional features are three sets of effective features for nearest neighbor classifier. They are good at describing the shape of the symbol but cannot capture the difference between two symbols if they are only different at a very small part.

2.2 Image-Based Math Symbol Retrieval

Math symbol retrieval can be done in two ways: (1) the symbol is recognized by OCR system and retrieval is done subsequently based on the symbol class label; (2) the retrieval is based on symbol image. Content-based math symbol image retrieval is a fairly new area of research and it can allow people to search a math symbol just using the image of itself. Marinai et al.[15] proposed an image-based math symbol retrieval method. They use Self Organizing Map (SOM) to cluster Shape Contexts (SC) extracted from each symbol into 30 classes. The feature vector is established based on the SC clusters by using 'bag-of-visual-words' model. Modified Angle Distance is used to calculate the difference between two feature vectors. Marinai et al.[16] use Earth Mover Distance (EMD) to achieve better accuracy. INFTYCDB-3 is used as dataset for experiment in.[15,16] 22,923 symbols are used for SOM clustering. They select 392 symbols randomly as the queries for retrieval from 259, 357 symbols. The best Area Under Curve (AUC) achieved in the system[16] is 33.13%.

3. INFTY DATASET

InftyCDB-1[10] is a ground-truthed math character and symbol image database. It contains 467 pages of 30 pure math articles in English. For each character, the ground truth, such as type, font, quality, link and so on, is attached manually. All the characters belong to words or formula in the database. InftyCDB-2 has the same structure as InftyCDB-1 and is a continuation of InftyCDB-1. InftyCDB-2 not only contains English articles but also French articles and German articles. InftyCDB-3 contains the alphanumeric characters and math symbols, but not word and math expression structure which are contained by InftyCDB-1. InftyCDB-3 is divided into two parts: InftyCDB-3-A and InftyCDB-3-B, and it is used as dataset for math symbol recognition[12] and retrieval evaluation.[16,20]

InftyCDB-3-A contains 188,752 symbols scanned in 400 dpi resolution while InftyCDB-3-B contains 70,637 symbols scanned in 600 dpi. InftyCDB-3-A contains 384 symbol classes while InftyCDB-3-B contains 275 classes; they share 266 classes. There are totally 393 symbol classes in InftyCDB-3-A and InftyCDB-3-B. Some classes only exist in one of the two datasets. There are four types of English fonts (Roman, Italic, Calligraphic and Blackboard Bold) and two types of numeric symbols (Roman and Italic). The number of classes is reduced to 279 in InftyCDB-3 if we ignore the fonts of English characters and numeric symbols. Table 2 gives the detailed analysis of InftyCDB-3.

For symbol recognition, just like,[12] we use InftyCDB-3-A for training and InftyCDB-3-B for testing. For symbol retrieval, we will use the whole InftyCDB-3 as the candidate library. There are 393 queries, one for each symbol class. Marinai et al.[16,20] also use the whole InftyCDB-3 as the candidate library, but only 392 queries. We are uncertain which symbol class was omitted.

Table 2: Symbol classes and instance numbers of InftyCDB-3. "Fonts Ignored" means English character and numeric symbol with different fonts are considered as belonging to the same class. "Distinct Class" means the symbol class which is in either InftyCDB-3-A or InftyCDB-3-B

	InftyCDB-3-A	InftyCDB-3-B	Total
Class (Fonts Ignored)	384 (270)	275 (190)	393 (279)
Instances	188,752	70,637	259,389
Distinct Class	118	9	127
Distinct Class Instances	8,318	31	8,349
Common Class Instances	180,434	70,606	251,040

4. SYMBOL RECOGNITION AND RETRIEVAL METRICS

Metrics used for recognition and retrieval are different but are similar in some aspects.

Recognition rate is usually used as the metric for symbol recognition. Recognition rate is defined as the percentage of correctly recognized symbols among total testing samples. Top-1 recognition rate is most often used. Many systems use Top-N or N-best (N >1) recognition rate. These systems provide N candidates for a given testing sample and the recognition is correct if the N candidates contain the target class. Combined with top-1 recognition rate, top-N recognition rate can give a precise measurement for the classifier, especially the cascading classifier. In cascading classifier, the result produced by the current level classifier will be the input for the next level classifier. Therefore, the top-N (N usually is 5, 10 and so on) recognition rate achieved by the current classifier is the best recognition rate the next classifier can achieve assuming the next classifier works on the filtered results produced by the current classifier.

For retrieval, Precision and Recall are used.[3] Precision is the fraction of retrieved instances that are relevant and Recall is the fraction of relevant instances that are retrieved.[21] Relevant instances can include inexact items. Final evaluation often requires that people provide relevance feedback by indicating whether returned results are relevant or irrelevant.[1] Top-1 Precision is defined as the percentage of the first retrieved result which is relevant and is highly related to the recognition rate produced by 1-Nearest Neighbor classifier. Top-N Precision is defined as the percentage of the first N retrieved results which are relevant and is related to Top-N recognition rate.

To measure the performance comprehensively, F-measure or F-score (the harmonic mean of Precision and Recall) is used.[4] $F = \frac{2PR}{P+R}$, where F is F-score, P is Precision and R is Recall. Getting higher F-measure requires reducing false positive and false negative simultaneously.

AUC of Precision and Recall is another important metric of retrieval. The Precision is plotted as a function of Recall. The value range for Recall is $[0, 1]$. Interpolation is always used to plot the Precision-Recall curve.[21] The area under the curve is computed by integral as the AUC. The larger the AUC is, the better performance the system achieves. The metric, Receiver Operating Characteristic (ROC), is similar to AUC. ROC curve is created by plotting the true positive rate against false positive rate. For retrieval, ROC can measure the system's ability to select the relevant instances as top candidates.

Another important measurement is Mean Average Precision (MAP). For a set of queries, MAP is the mean of the average precision scores for each query. The average precision scores are the average of the top-1, top-2, ..., top-N precisions. MAP is inherently biased toward early returns. The earlier the relevant symbol is retrieved, the higher value MAP will be.

5. METHODOLOGY

5.1 Turning Functions

Arkin et al.[7] proposed a metric based on Turning Function (TF) to compare polygonal shapes. Assuming $\Theta_A(s)$ represents the Turning Function of polygon A, $\Theta_A(s)$ is a function of the arclength s and measures the counterclockwise tangent angle measured from the starting point O on A's boundary.[7] The main idea of Turning Function is to transform a 2-D shape into a 1-D vector for comparing while preserve its shape information. Many systems in medical imaging[22, 23] or CBIR[24, 25] show that Turning Function is good at distinguish simple polygonal shapes. Our feature is also based on Turning Function and can distinguish the outer contours of math symbols very well. We use Turning Function for math symbol recognition and retrieval.

5.2 Preprocessing

Contour Extraction: Getting Turning Function just needs the contour of each symbol. The image of each symbol is binarized and boundary of the symbol is defined. We keep all the pixels along the contour counterclockwise but discard all other pixels. The starting point determines the order of Turning Function. It is important to choose the starting point in the same way for all the symbols. We choose the pixel at the top of the leftmost column on the contour as the starting point. This selection will be sensitive to rotation or noise. We will use an alignment method to resolve the issue.

For symbols consist of more than one CCs, we extract the boundaries of each CC and concatenate their contours. The CC contours are queued by CCs' centroid positions from top to bottom and left to right, then concatenated. This method will introduce error when test images have large rotational angles which is very rare.

Smoothing: Extracted contour above is not smooth. In order to reduce noise, we use a sliding window with size five for smoothing. For each pixel, its coordinates after the smoothing are the average of the coordinates of five pixels (the current one and its four neighbors).

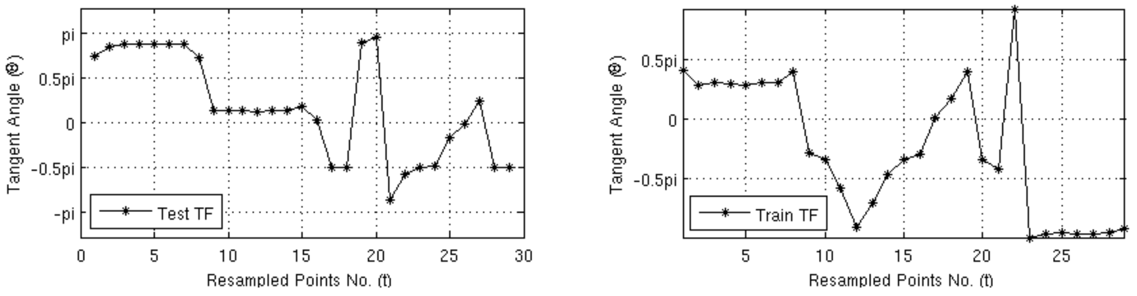
Resampling: Arkin et al.[7] used the original vertices of polygon boundary to calculate the Turning Function. To deal with noise and ambiguous corners better, we resample 30 equally spaced points for each symbol's boundary, which is often used in handwritten math symbol recognition.[26] This also makes comparing the Turning Functions of two polygon boundaries easier.

5.3 Calculating Turning Function

In order to get the Turning Function, we compute the 1st order differences (δx and δy) of the coordinates of the pixels along the symbol's contour at first, and then the Turning Function can be computed as:

$$\Theta(s) = arctan(\frac{\delta y(s)}{\delta x(s)}).$$

We use four-quadrant arctan to compute $\Theta(s)$, and signs of δx and δy are used to determine the quadrant. Therefore, the value range of $\Theta(s)$ is $[-\pi, \pi]$ (as shown in Figure 1) while the value range of regular arctan is $[-\pi/2, \pi/2]$. The current Turning Functions have the same length (30 points for each symbol) and their value range is $[-\pi, \pi]$. There are two problems with the current Turning Functions: (1) $[-\pi, \pi]$ is not enough to contain all the turn function value; (2) there is no alignment to deal with rotation and changing of starting point.

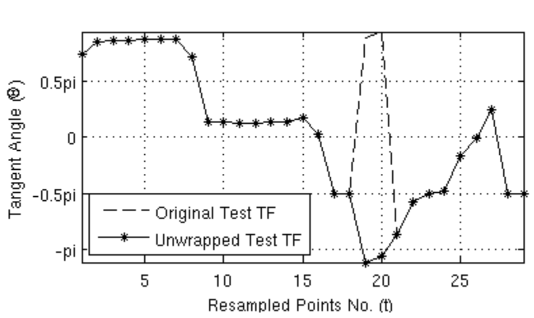


(a) Turning function of test sample (b) Turning function of train sample

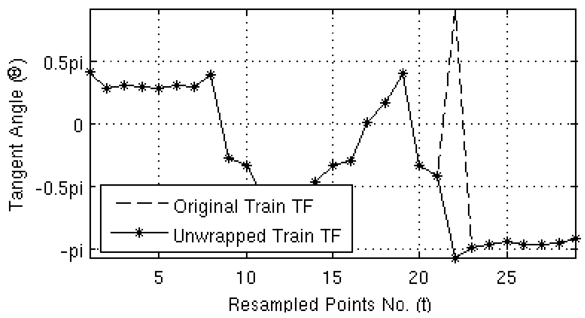
Figure 1: Turning functions of test sample (Test TF) and train sample (Train TF)

Unwrap: We use a unwrap process to deal with the value range insufficiency. For a convex polygon A, $\Theta_A(s)$ is monotone increasing.[7] The value range of $\Theta_A(s)$ is $[v, v + 2\pi]$ where v is the Turning Function value at the starting point O. The value of $\Theta_A(s)$ can be arbitrarily large for a non-convex polygon. As we mentioned above, tangent angle's value range is restricted to $[-\pi, \pi]$ by the method we used. Angle value jump will take place while the absolute value of $\Theta_A(s)$ is larger than π. For example, while previous point have an angle of 0.9π and current point have an angle of 1.1π, the increment should be 0.2π. But current point is forced to be -0.9π because of the value range, meanwhile increment becomes -1.8π. To deal with this problem, unwrap method is introduced. Unwrap method is well-known in signal processing while dealing with phase angles. The difference between current point and previous point is computed, if the absolute value of jump is greater than π, then the current point value will add or subtract 2π to achieve the smallest value jump. After unwrap process, there is no limit for the Turning Function's value range. Figure 2 shows an example of unwrapped testing and training Turning Function (TF).

Alignment: We propose an alignment to deal with rotation and changing of starting point. Turning function is invariant under translation and scaling but dependent to the rotation and changing of starting point. Rotation of A counterclockwise by θ causes a vertical shift of $\Theta_A(s)$ and the new Turning Function is $\Theta_A(s) + \theta$ (as shown in Figure 3 (a)). Changing the location of staring point O along the perimeter of A by an amount t results in a horizontal shift of $\Theta_A(s)$ and the new Turning Function is $\Theta_A(s + t)$ (as shown in Figure 3 (b)). In practice, Turning Function usually has the horizontal and vertical shift simultaneously because of the rotation and difference of starting point. For two Turning Functions, we propose a method to find their smallest distance by searching horizontally and vertically. The time complexity of the searching method is $O(m^2)$, where m is the length of Turning Function (30 in our system). Firstly, we concatenate two copies of one Turning Function (Turning Function of test symbol in our system) to form a new function with double length. We call it duplicated Turning Function, as shown in Figure 4. Then unwrap process is performed upon both the duplicated Turning Function and the other Turning Function (Turning Function of training sample). Secondly, we will extract 30



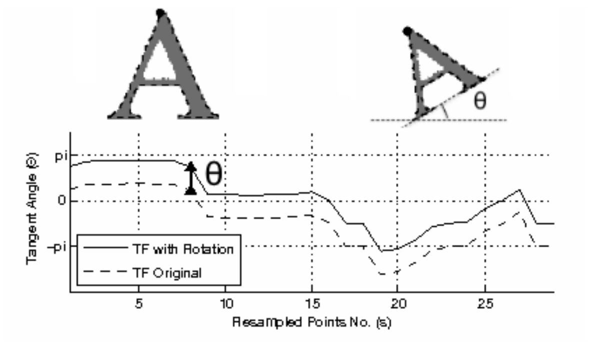
(a) Unwrapped Test TF



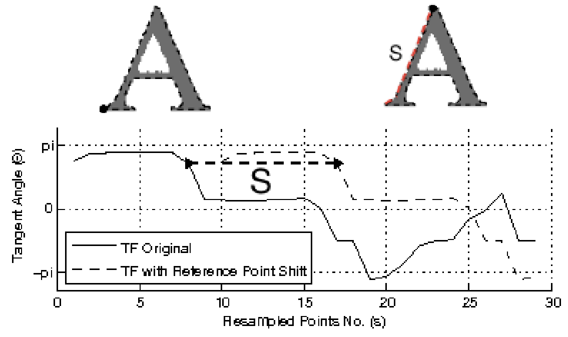
(b) Unwrapped Train TF

Figure 2: Unwrapped Test TF and Train TF

Turning Functions whose length is 30 by sliding along the horizontal axis of the duplicated Turning Function. These 30 Turning Functions represent all the possible Turning Functions with different starting point. Computing the Sum of Square Distance between each of the 30 Turning Functions and the Turning Function of training sample could solve the horizontal shift problem caused by the changing of the starting point. Thirdly, before calculating the distance between two Turning Functions, each of the two Turning Functions will be substracted by the average angle in each Turning Function and this could mitigate the vertical shift caused by rotation. Figure 5 shows an example of matching Unwrapped Train TF and duplicated Unwrapped Test TF.

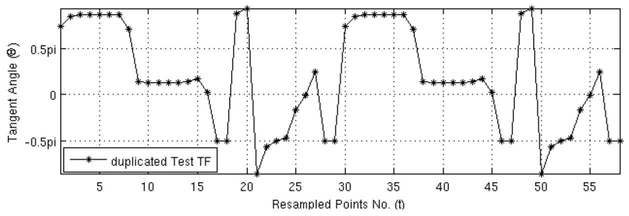


(a) Vertical shift

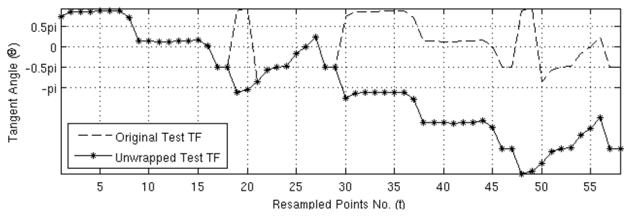


(b) Horizontal shift

Figure 3: Vertical and horizontal shifts in Turning Function (TF)



(a) Duplicated Test TF



(b) Unwrapped duplicated Test TF

Figure 4: Duplicated Test TF and unwrapped duplicated Test TF

The main disadvantage of Turning Function is that it discards the symbol's internal shape information which is useful for symbol recognition. Grid features perform very well for noise free dataset. We would like to use Grid Features to capture these information discarded by Turning Function to improve recognition rate. Here we compute using 5×5 grid cells. Then each of the five most similar cells are divided by finer 5×5 grid to get more detailed comparison. It is like a quad-tree style segmentation of images, but 5×5 grid is used instead of 2×2. The distances between those smaller cells are weighted, so when combined they still get the same weights as other big cells. By using this quad-tree style finer grid cells and catching detailed information in

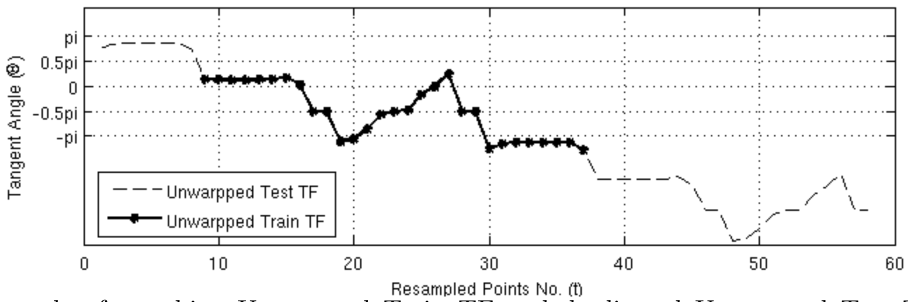


Figure 5: An example of matching Unwrapped Train TF and duplicated Unwrapped Test TF (Getting the minimum SSD between Unwrapped Train TF and Unwrapped Test TF)

similar regions, we would get a improved recognition rate. Grid features are very sensitive to rotation and noise. Although Turning Function is also sensitive to the rotation. We can estimate the rotation angle by using Turning Functions. By comparing the vertical displacement between the Turning Functions of test symbol and training symbol, the rotational angle can be calculated directly. In our system, given the test symbol, we find 50 symbols in the training set with smallest distance to the test symbol based on Turning Function. In this way, we can get 50 rotational angles at most. Then we rotate the image back to its original position by using these rotational angles. At last, Grid Features are used to do the classification among the top-50 candidates produced by Turning Function. Here we call this Turn Function Alignment and Grid Feature Distance (TFA-GVD) method.

6. MATHEMATICAL SYMBOL RECOGNITION EXPERIMENT

We use InftyCDB-3-A for training and InftyCDB-3-B for testing. The use of dataset is the same as Malon et. al.[12] 1-Nearest Neighbor classifier is used based on Turning Function and Grid Features. For two symbols, SSD is calculated as the distance between them using the two feature vectors. Top-1 recognition rate is chosen as the metric. Figure 6 shows the comparison of recognition rates by using different features.

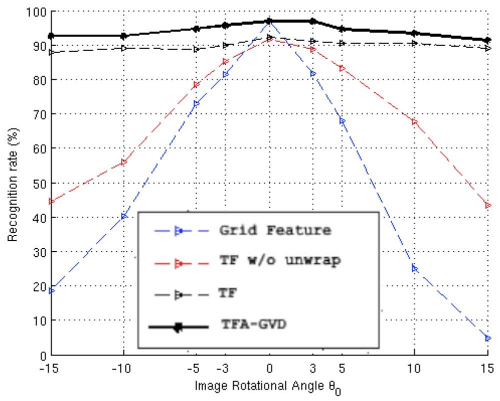


Figure 6: Comparison of recognition rates and robustness against rotation by using different features.

Dataset InftyCDB-3 is clean and noise free. First, we just use Grid Features. Table 3 shows that Grid Features achieve 97.04% recognition rate using quad-tree style finer grid comparison as discussed before, which is higher than using pure 5×5 grid cells by nearly 1%.

The pure Infty engine[11] gets 96.10% accuracy. Malon et al.[12] adds binary support vector machines (SVM) classifier to the pure Infty engine and gets 97.70% recognition rate.

Secondly, Turning Function is tested. We did experiments with and without unwrap and alignment to show the necessity of these processes. Experiment result shows that Turning Function with unwrap and alignment perform better than without. With unwrap and alignment, Turning Function's recognition rate is still lower than

Table 3: Recognition Rate and runtime using Grid Feature, Turning Function, and Turning Function alignment plus Grid Feature Distance method (TFA-GVD) while rotating symbol images.

	Grid Feature	TF (w/o unwrap)	TF	TFA-GVD
Rotation Angle	Recognition Rate(%)			
0°	**97.04**	91.65	92.32	96.90
3(-3)°	81.71(81.54)	88.75(85.09)	91.20(90.04)	**96.84(95.62)**
5(-5)°	67.89(72.83)	83.22(78.57)	90.44(88.84)	**94.58(94.66)**
10(-10)°	24.99(40.27)	67.80(55.96)	90.41(88.89)	**93.41(92.55)**
15(-15)°	4.89(18.57)	43.28(44.65)	89.09(87.88)	**91.35(92.67)**
RunTime	02:02:26	02:06:20	03:29:27	09:31:18

Grid Features'. The reason is that Turning Function does not consider the symbol's internal shape information. For example, 'O' and 'Θ' are hard to be distinguished by Turning Function.

Compared to Grid Features, Turning Function is more robust against rotation. To test this, we perform experiment with rotated images. The training images are not modified, while testing images are rotated by 3, 5, 10 and 15 degrees clockwise and counterclockwise. Experiment result shows that Turning Function with unwrap and alignment is much more robust against rotation than Grid Features. The recognition rate of Grid Features drops from 97.04% (no rotation) to 4.89% (15 degrees rotation counterclockwise). For Turning Function, the decrease of recognition rate caused by rotation is much smaller.

By utilizing TFA-GVD method, we manage to take advantages from both methods. The Turning Functions are used to contour rotation transformation by estimating angle displacement, while Grid Features are used to get best recognition rate once the image is rotated back to its original position. The best recognition rate reaches 96.90% with no rotation and remains 91.35% with 15 degrees rotation counterclockwise.

Grid feature and Turning Functions without unwrap/alignment use similar runtime. However, Turning Functions with unwrap and alignment cost more time because of the searching for alignment between test and train functions. The searching is done within time $O(m^2)$ as we discussed before. The TFA-GVD will cost more time, as we have to firstly find the most similar 50 samples using Turning Functions and rotate the image accordingly, then use the Grid Feature to get the final classification result. Here, all experiments are processed using InftyCDB-3-A (contains 188,752 samples) as training dataset and InftyCDB-3-B (70, 637 samples) as testing dataset. Because of the exhaustive search, nearest neighbor classifier is slow. Clustering training dataset would significantly reduce the runtime. Experiment shows sorted histogram of Turning Function shows great potential for data clustering. We would to test this in the future work.

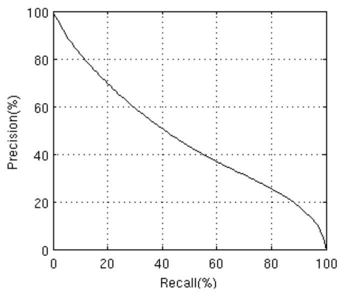
The experiments are run on our Linux server which has 24 CPU cores, 96 Gb of RAM. Matlab software is utilized, parallel computing with 8 labs are used to reduce computation time. The average total runtime of each method is provided in Table 3.

7. MATHEMATICAL SYMBOL RETRIEVAL EXPERIMENT

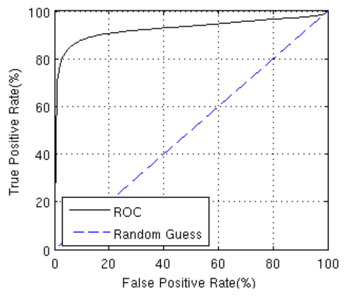
The recognition experiment results show that unwrapped Turning Function with alignment and Grid Features perform the best. We use the same features and metric for the math symbol retrieval experiment. Turning function is used to adjust the rotation and change of starting point between query symbol and target symbols. After the alignment, Grid Features are used to calculate the distance based on SSD between query symbol and target symbols. The retrieval results are ranked based on this distance.

For the retrieval experiment, we use the whole InftyCDB-3 as the candidate library and 393 queries, one for each symbol class. Marinai et al.[16, 20] also use the whole InftyCDB-3 as the candidate library, but only 392 queries. We believe that they chose a query for each symbol class while ignored one symbol class because it is too small. But we are uncertain which symbol class was omitted.

The whole retrieval experiment for 393 queries takes 432 seconds. Top-25 precision, Top-50 precision, Mean Average Precision and AUC of Precision/Recall are computed and shown in Table 4. Marinai's Shape Context based method[16] achieved (33.13%) AUC, our AUC is 47.27%. The standard deviations are small, which means our system performs consistently for different kinds of symbols. To evaluate the average performance of our retrieval algorithm, we plot Precision/Recall curve (Figure 7 (a)) and Receiver Operating Characteristic curve (Figure 7 (b)). From Figure 7 (a), we can find that our Top-1 precision is almost 100%. Precision drop rate is roughy constant with a tail at the end. Figure 7 (b) shows that ROC curve is very close to the point (0,1) at one time which means our retrieval system can find the relevant symbols very early while discarding irrelevant symbols.



(a) Precision/Recall curve

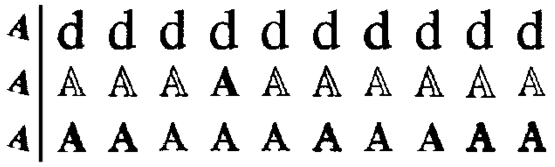


(b) ROC curve

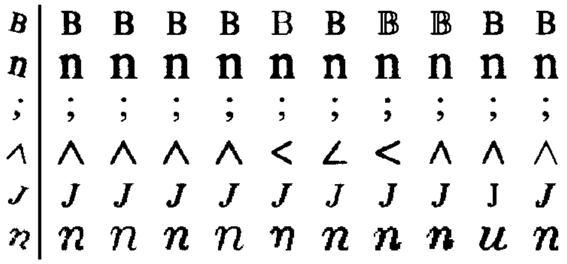
Figure 7: Precision/Recall curve and Receiver Operating Characteristic (ROC) curve

Table 4: Mean (μ) and standard deviations (σ) over 393 queries' Top 25, Top 50 precision, Mean Average Precision (MAP), and AUC of Precision/Recall.

	Top 25	TOP 50	MAP	$\text{AUC}_{\text{P/r}}$
$\mu(\sigma)$	76.35%	71.20%	6.43%	47.27%
	(3.16%)	(3.31%)	(1.29%)	(2.73%)



(a) Symbol image 'A' retrieval with different methods.



(b) Different symbols with best method.

Figure 8: Top-10 retrieval results of queries. Left-most column shows the query symbol, here all queries are rotated 15 degrees clockwise to test retrieval systems' robustness. The following images in each row shows corresponding retrieval result. (a) 1st row, Grid Feature; 2nd row, Turning Function; 3rd row, TFA-GVD. (b)Different queries using TFA-GVD.

The example of Top-10 retrieval results are shown in Figure 8(a). Figure shows Top-10 results for testing symbol A with rotational angle equals 15 degrees clockwise. The first row shows the SSD ranking using Grid Feature Vector result. Here we could see, though Grid Feature performs outstandingly for noise free images (due to the cleanness of Infty dataset), it gives totally wrong Top-10 retrieval result for a image with 15 degree of rotation. The sensitivity to noise and deformation is the main weakness of Grid based features.

Meanwhile, the second row shows the Top-10 results using Turning Functions. The retrieved symbols are mostly Blackboard Bold font instead of desired Roman font. Though the symbols are regarded as irrelevant if we follow the INFTY symbol coding system, we get character symbol 'A' correct. The reason why we are not getting the Roman A as we were expecting is missing of internal information of shapes.

The third row shows the Top-10 retrieval result while using Turning Function to estimate rotational angle and Grid Feature to measure distance for restored test images. Here we could see, by compensating rotations using Turning Functions alignment and catching internal information using Grid Features, we successfully retrieved exactly correct symbols in the Top-10.

Some Top-10 retrieval examples are given in Figure 8(b), where Turning Function Alignment and Grid Feature Distances are utilized (TFA-GVD). Here we could see, some images retrieved are with same character but different fonts, e. g. symbol 'B', 'n' and 'J'. Although, they are considered as false positive in our evaluation system due to their different INFTY symbol codes, human would consider they are relevant. In information retrieval, final judge of relevance could only be made correctly by human. To compare with previous work,[16] we follow the INFTY coding system and assume different fonts are irrelevant symbols. But if a human expert is asked to evaluate the system, he/she will give a higher score of performance.

We see for symbols consist of two CCs, e. g. ';', we could still manage to get the correct results. It supports our contour concatenation method is good in this case.

We could see some symbols retrieved are different (e. g. we get symbol '$<$' from query '$\wedge$', and symbol 'η' from 'u'), but only with a rotational transform. If a largest possible rotational angle of noise could be measured (usually small), we could limit the Turning Function Alignment between such threshold values and avoid false positive with rotational similarities, like '9' and '6'.

8. CONCLUSION

The paper presents a system for recognition and retrieval of isolated offline printed mathematical symbols for the first time. The system is based on nearest neighbor analysis and we use two features: modified Turning Function and Grid Features. Both Turning Function and Grid Features are sensitive to rotation while Turning Function is also sensitive to changing of starting point. We modify the Turning Function by using an unwrap technique and an alignment method. The modified Turning Function can be used to get the rotation angle and difference of starting points of two symbol images. Based on the rotation angle, we can rotate the image to its original position. Grid features are extracted and used to calculate the distance based on SSD. The experiment results show the modified Turning Function can deal with the rotation and changing of starting point very well. Compared to results achieved by previous work, our system's performance is quite competitive.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1016815. We thank Simone Marinai for helpful discussions about Precision/Recall AUC metrics.

REFERENCES

1. R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *International Journal on Document Analysis and Recognition* , pp. 1–27, (available online), Springer, 2011.
2. K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: a survey," *International Journal on Document Analysis and Recognition* **3**, pp. 3–15, Aug. 2000.
3. A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**, pp. 1349–1380, December 2000.
4. R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Comput. Surv.* **40**, pp. 5:1–5:60, May 2008.
5. R. Datta, J. Li, and J. Z. Wang, "Content-based image retrieval: approaches and trends of the new age," *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval* , pp. 253–262, 2005.

6. R. Zanibbi and L. Yu, "Math spotting: Retrieving math in technical documents using handwritten query images," in *International Conference on Document Analysis and Recognition*, pp. 446 –451, sept. 2011.
7. E. Arkin, L. Chew, D. Huttenlocher, K. Kedem, and J. Mitchell, "An efficiently computable metric for comparing polygonal shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence* **13**, pp. 209 –216, mar 1991.
8. R. Fateman, T. Tokuyasu, B. P. Berman, and N. Mitchell, "Optical character recognition and parsing of typeset mathematics," *Journal of Visual Communication and Image Representation* **7**, pp. 2–15, 1996.
9. F. Alvaro, J.-A. Sanchez, and J. Benedi, "Recognition of printed mathematical expressions using two-dimensional stochastic context-free grammars," in *International Conference on Document Analysis and Recognition*, pp. 1225 –1229, sept. 2011.
10. M. Suzuki, S. Uchida, and A. Nomura, "A ground-truthed mathematical character and symbol image database," in *International Conference on Document Analysis and Recognition*, pp. 675 – 679, Sept 2005.
11. M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori, "Infty: an integrated ocr system for mathematical documents," in *ACM symposium on Document engineering*, pp. 95–104, (New York, NY, USA), 2003.
12. C. Malon, S. Uchida, and M. Suzuki, "Mathematical symbol recognition with support vector machines," *Pattern Recognition Letters* **29**, pp. 1326–1332, July 2008.
13. U. Garain, B. Chaudhuri, and R. Ghosh, "A multiple-classifier system for recognition of printed mathematical symbols," in *Proceedings of the 17th International Conference on Pattern Recognition*, **1**, pp. 380 – 383, Aug. 2004.
14. F. Alvaro and J. Sanchez, "Comparing several techniques for offline recognition of printed mathematical symbols," in *20th International Conference on Pattern Recognition*, pp. 1953 –1956, Aug. 2010.
15. S. Marinai, B. Miotti, and G. Soda, "Mathematical symbol indexing using topologically ordered clusters of shape contexts," in *10th International Conference on Document Analysis and Recognition*, pp. 1041–1045, (Washington, DC, USA), 2009.
16. S. Marinai, B. Miotti, and G. Soda, "Using earth mover's distance in the bag-of-visual-words model for mathematical symbol retrieval," in *International Conference on Document Analysis and Recognition*, pp. 1309 –1313, Sept. 2011.
17. U. Garain and B. Chaudhuri, "A corpus for ocr research on mathematical expressions," *International Journal on Document Analysis and Recognition* **7**, pp. 241–259, Sept. 2005.
18. R. Plamondon and S. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, pp. 63 –84, Jan 2000.
19. S. Espana-Boquera, M. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, "Improving offline handwritten text recognition with hybrid HMM/ANN models," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **33**, pp. 767 –779, April 2011.
20. S. Marinai, B. Miotti, and G. Soda, "Mathematical symbol indexing," in *AI*IA 2009: Emergent Perspectives in Artificial Intelligence*, **5883**, pp. 102–111, Springer Berlin / Heidelberg, 2009.
21. G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc., New York, NY, USA, 1986.
22. S. Shiffman, G. Rubin, and S. Napel, "Medical image segmentation using analysis of isolable-contour maps," *IEEE Transactions on Medical Imaging* **19**, pp. 1064 –1074, Nov. 2000.
23. D. Guliato, R. Rangayyan, J. Carvalho, and S. Santiago, "Polygonal modeling of contours of breast tumors with the preservation of spicules," *IEEE Transactions on Biomedical Engineering* **55**, pp. 14 –20, Jan. 2008.
24. A.-S. Chia, S. Rahardja, D. Rajan, and M. Leung, "Structural descriptors for category level object detection," *IEEE Transactions on Multimedia* **11**, pp. 1407 –1421, Dec. 2009.
25. C. Erdem, B. Sankur, and A. Tekalp, "Performance measures for video object segmentation and tracking," *IEEE Transactions on Image Processing* **13**, pp. 937 –951, July 2004.
26. L. Hu and R. Zanibbi, "Hmm-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature," in *International Conference on Document Analysis and Recognition*, pp. 457 –462, sept. 2011.