Chinese Character Images Recognition and Its Application in Mobile Platform

Gang Gu, Jiangqin Wu, Tianjiao Mao and Pengcheng Gao
Digital Media Computing and Design Lab (DCD), Zhejiang University, 38 Zhejiang University Road, Hangzhou, China

Keywords: GIST-SIFT-SSC, Image Understanding, Chinese Characters Recognition, Mobile Platform.

Abstract: Chinese characters are profound and polysemantic. Reading a Chinese character is a procedure of image understanding, if the Chinese character is captured as an image. Due to the complexity of structure and plenty of Chinese characters, there always exist some unfamiliar characters when reading books, so it would be great if a tool is provided to help users understand the meaning of unknown characters. We propose a method that combines global and local features(i.e., GIST and SIFT features) to recognize the Chinese character images captured from mobile camera. Three schemes are investigated based on practical considerations. Firstly,the so-called GIST and SIFT descriptors extracted from Chinese character images are adopted purely as features. Then filter the SIFT feature points of similar Chinese character images based on GIST feature. Finally, compress the storage of GIST and SIFT descriptors to accommodate mobile platform with Similarity Sensitive Coding(SSC) algorithm. At the stage of recognition, the top 2k Chinese characters are firstly obtained by hamming distance in GIST feature space, then reorder the selected characters as final result by SIFT feature. We build an Android app that implements the recognition algorithm. Experiment shows satisfying recognition results of our proposed application compared to other Android apps.

1 INTRODUCTION

Chinese character is the basic unit of Chinese language. Meanwhile, it is one of the finest Chinese art forms and the most inseparable parts of Chinese history. Its delicate aesthetic effects are generally considered to be unique among all Chinese arts. However, the Chinese characters are complicated due to its quantity and structure. Fig. 1 shows the different typefaces of a Chinese character. Therefore, when people, Chinese or foreigners, read Chinese books, they may meet some unfamiliar characters. It will be quite beneficial to them for reading Chinese books, if there exists a tool that could recognize Chinese character images.

Chinese character images recognition on mobile platform is a valuable and challenging task in the area of image understanding. It is quite convenient to use the phone camera to capture images of unfamiliar Chinese characters and recognize them. Although there are many mobile recognition applications, the accuracy is not very high. There is still much more space for further development. The recognition algorithm applied in mobile platform still poses difficult challenges. The reasons are manifold: (a) The structure of a Chinese character can be very complicated. (b)The massive Chinese characters engenders considerable computation and storage cost. (c) what features to represent the character image and how to compress the storage of features to accommodate the mobile platform. In addition, the feature extraction and the recognizing procedure are two important issues to be considered.

For the feature extraction, there mainly exist two categories of feature for image representation: global feature and local feature. Zhuang *et al.* (Zhuang et al., 2005) extracted shape feature from character images

Figure 1: The different typeface of a Chinese character.

Gu, G., Wu, J., Mao, T. and Gao, P.
Chinese Character Images Recognition and Its Application in Mobile Platform.
DOI: 10.5220/0005679703110318
In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016) - Volume 3: VISAPP*, pages 313-320
ISBN: 978-989-758-175-5

for content-based character retrievalbut the dimension of shape feature is not unified and it is too large for mobile devices. One of the widely used global features is GIST descriptor. GIST descriptor was initially proposed in (Oliva and Torralba, 2001) and widely used in content-based image retrieval (Barbu, 2009; Pengcheng et al., 2014). It is proved that GIST descriptor can describe an image by a set of low dimensions, such as naturalness, openness, roughness, expansion, ruggedness, which are also very distinguishable when describing Chinese characters. Y. Lin *et al*. used GIST descriptor to represent Chinese calligraphy characters in (Lin et al., 2013)to recognize large scale Chinese calligraphic characters.

Beside the global features, local features are also adopted in many applications. SIFT descriptor was created by David Lowe in (Lowe, 2004) to represent the image. which has been proved to be the one of the best local feature descriptors (Mikolajczyk K, 2005). It is widely used for object recognition (Rosenberg and Dershowitz, 2012; Zhen-Yan, 2014). SIFT feature is also used in Chinese recogniton(Zhang Z, 2009; Jin et al., 2009). Chen *et al*. applied SIFT to Chinese character recognition in license plate and achieved great performance. Mao *et al*. used SIFT feature to distinguish the style calligraphy words and achieved high accuracy (Tianjiao et al., 2013). However, the number of SIFT keypoints in one character image often varies from tens to hundreds. So the library of SIFT points is much larger and the recognition will cost much more computing power.

With regard to the recognizing procedure, Shaknarovich and Darell (Shakhnarovich G, 2003) proposed a Similarity Sensitive Coding(SSC) algorithm extending Locality Sensitive Hashing (LSH), whose key contribution is creating a new binary ($+1$ and -1) feature space mapped by hash functions learned from training set. To speed up matching and reduce memory consumption, Calonder M (Calonder M, 2010) used binary (0 and 1) strings as a feature descriptor. The descriptor similarity can be evaluated by using the Hamming distance, which achieves great recognition performance. The Hamming distance can be computed very efficiently with a bitwise XOR operation followed by a bit count, so the high-dimensional search is very fast. (Calonder M, 2010) also confirmed the trend (Shakhnarovich G, 2003) of moving from the Euclidean to the Hamming distance in matching procedure.

The main contribution of this paper is to propose a recognition algorithm for the Chinese character images captured from camera, which is based on GIST and SIFT features. It ensures the high accuracy and great performance on mobile platform. Meanwhile, it compress the storage of feature vectors by SIFT keypoints filtering and SSC encoding. The two methods of compressing greatly reduce the computational cost. Moreover, the recognition algorithm decreases the errors caused by SSC encoding, because the SIFT keypoints are filtered before SSC encoding, and the preponderance of two features will make up for it.

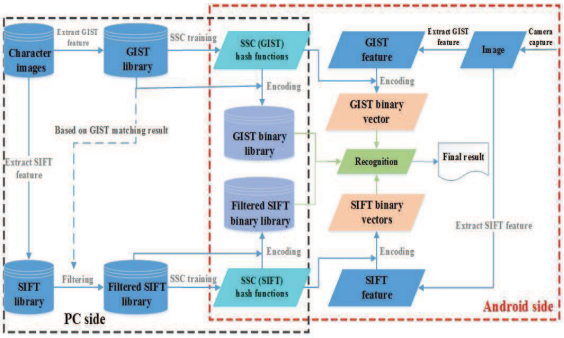

Figure 2: Overall development flow and architecture.

The remainder of the paper is organized as follows. In Section 2, we give a system overview of our proposed approach. Section 3 describes the methods of image representation and recognition. Then section 4 presents the system implementation and the experimental results. Finally we conclude the paper in section 5.

2 SYSTEM OVERVIEW

An overall system development flow and architecture is illustrated in Fig. 2. For the mobile Chinese characters recognition platform, we first build the Chinese character images library of Windows system installed Chinese font (Hei and Song typeface most used in Chinese books), then extract GIST and SIFT feature from the Chinese character images to create GIST and SIFT feature library. After that, we filter SIFT keypoints of similar character images to reduce the computational cost and compress the storage of the SIFT library. During the training stage of SSC, images of the same character are treated as positive pairs and otherwise as negative pairs for GIST feature training, positive or negative keypoints measured by distance for SIFT feature training. Two hash functions obtained from the training will map feature vectors to binary codes respectively. After that, the similarity of Chinese characters can be measured by hamming-distance and the storage of feature library can be extremely compressed. All the previous processes can be done off-line on PC. At the recognition stage, the feature vectors are first extracted from the unknown character image captured from the phone

camera, then encode GIST and SIFT vectors to binary codes by hash functions, The final result will be computed by recognition algorithm.

3 IMAGE REPRESENTATION AND RECOGNITION

Three steps have to be done to implement the recognition of images. Firstly, it is important to process images collected by different ways and select some proper images to build the dataset. Then extract the image features from the dataset to create the feature library, which is the basis of recognition algorithm. The third step is to design an efficient algorithm to recognize images from the feature library. The following parts will unfold in this way.

3.1 GIST-SIFT based Image Representation

In this section, we will discuss the method to build images library and choose great features to represent Chinese characters images. We use Windows system installed Chinese font to build the images library of Chinese characters, in which text segment technology is applied to get each character image from PDF files. There are two types of fonts frequently used in printed Chinese materials, namely Song typeface and Hei typeface. Each typeface contains 20780 Chinese characters according to Xinhua dictionary, which is the most popular dictionary in China. After building images library, the next step is to select proper image feature.

Representation of character images is significant in Chinese character recognition, because the difference between Chinese characters is so delicate. So the image representation needs sophisticated descriptors. In this paper, we use GIST descriptor to develop the low dimensional representation of the Chinese characters. GIST descriptor is computed by three steps:

- Convolve the image with 24 Gabor filters at 3 scales, 8 orientations, producing 24 feature maps of the same size of the input image.
- Divide each feature map into 9 regions (by a 3×3 grids instead of 4×4(Ni et al., 2008)), and then average the feature values within each region.
- Concatenate the 16 averaged values of all 24 feature maps, resulting in a $3 \times 8 \times 9 = 216$ GIST descriptor.

Intuitively, GIST summarizes the gradient information (scales and orientations) for different parts of an image, which provides a rough description (the gist) of the scene. In our method, the image is first compressed to 32×32 pixels, because printed Chinese characters have a relatively limited form, 32×32 pixels is big enough for recognition. And also we divide the images into 3×3 sub images instead of 4×4 grids or more for the same reason. After this process, the character image is represented by a $3 \times 8 \times 9 = 216$ dimensional vector. Fig. 3 shows the gist descriptors of Chinese character image under different girds. It implies that 3×3 grids of GIST feature is capable for the unique feature of each Chinese character image.

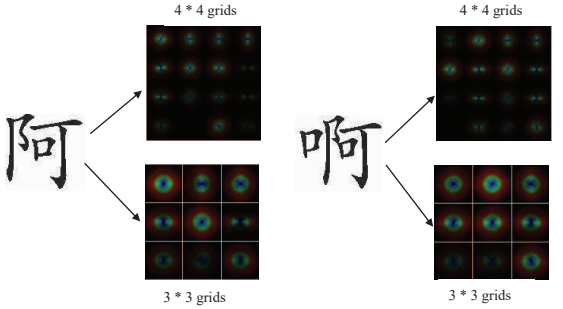

Figure 3: Chinese character images and their corresponding GIST descriptors.

Meanwhile, we use SIFT feature to represent the local feature of the characters images. SIFT descriptor is based on the gradient distribution information within the scale region of pre-detected keypoint, and constructed from a 3D histogram (gradient locations and orientation).It is a 128-dimension vector standing for the bins of the oriented gradient histogram (Lowe, 2004). We use Scale Invariant Feature Transform (SIFT) algorithm (Lowe, 2004) to extract SIFT feature from Chinese character images and get SIFT feature vectors of each character. A high level overview of the extraction process of a single SIFT descriptor is as follows:

- Given a center, the gradients and their magnitudes are computed in an area around the center of the descriptor, then the magnitudes are weighted by a Gaussian window.
- The area around the descriptor is split into 4×4 spatial bins. The size of each spatial bin (in pixels) is the scale of the descriptor.
- For each spatial bin, a histogram of 8 orientation bins, weighted by the magnitudes is calculated. The area that is taken into account is defined by the scale of the descriptor and a magnification factor. This results in a $4 \times 4 \times 8$=128-dimensional descriptor.

Finally, we extract GIST and SIFT feature of the two typefaces images and build GIST and SIFT li-

brary. Further more, GIST vector is represented by a float vector, that means each GIST vector takes $4 \times 216 = 864$ bytes. The number of SIFT vectors of each character image is from ten to hundreds, and each SIFT vector being represented by a integer vector takes $4 \times 128 = 512$ bytes. After feature extracting, the GIST library will take $41560 \times 864 = 35907840$ bytes (34.24 MB) and SIFT feature needs $4320626 \times 512 = 2212160512$ bytes (2.06 GB) to store, which is a much bigger memory and storage cost for mobile devices. We used two methods to make them smaller to accommodate the limitation of mobile platform.

3.2 SIFT Keypoints Filtering

For SIFT feature, the similarity of Chinese characters is measured by the number of feature points matched between their images, so the similar feature points in similar character images will cost the extra computational capacity(Fig. 4 shows the SIFT feature points in similar character images). We will filter the similar feature points and reserve unique points in each character images. To remove similar feature points in

Figure 4: Samples of SIFT keypoints in similar characters *(these similar characters images obtained by GIST feature)*.

each image, we use KDTree-based filtering method. The main idea is that if the Euclid distance of SIFT feature points between similar character images is less than the threshold, these keypoints will be regarded as similar SIFT feature points to be removed. For each GIST vector in GIST library, find its k nearest neighbors as similar Chinese character images. A pseudocode description for the filtering process is given in Algorithm 1. As a result, the filtered SIFT feature library takes 317.16 MB. However, the library is also too large for mobile platform, we need to compress it.

3.3 SSC based Feature Vector Encoding

Firstly, we give a brief overview of Similarity Sensitive Coding(SSC)(Shakhnarovich G, 2003), the target of that algorithm is to learn an similarity mapping function, which is faithful to a task-specific similarity:

$$H : x \rightarrow [\alpha_1 h_1(x), ..., \alpha_m h_m(x)] \qquad (1)$$

Algorithm 1: SIFTLibraryFilter(L_1,L_2,k,*threshold*): Filter similar keypoints in similar character images.

Input:
L_1 : The GIST feature library of the typeface.
L_2 : The SIFT feature library of the same typeface, every feature $Sift_i$ in the library has SIFT feature vectors $v_1, v_2, ..., v_m (m > 0)$, each vector in which has responding valid value $valid_m$ default value is 1.
k : The number of nearest neighbors to search.
threshold : The max Euclid distance between a pair of similar keypoints.
Output:
R :The filtered SIFT feature library.

```
Let C = C_1,...,C_n be the Chinese characters in L_1;
For each GIST feature vector T_i (i = 1,2,...,n) in L_1;Let P_i = {k similar Chinese characters with C_i computed from nearest feature vectors of T_i by KDTree};
For each SIFT feature Sift_i in L_2:
    For each v_j in SIFT feature Sift_i:
        flag = 0;
        For each v_ψ in Sift_p of C_p in P_i:
            dist= Euclid distance between v_j and v_ψ;
            If dist ≤ threshold:
                Let valid_p[ψ] = 0 and flag = 1;
        If flag == 1:Let valid_i[j] = 0;
For each SIFT feature Sift_i in L_2:
    For each v_j in SIFT feature Sift_i:
        If valid_i[j] == 0:remove v_j from Sift_i;
```

where x is the original input, α_i is weights, each dimension m is produced by a function h_m, parametrized by a projection:

$$h(x; f, T) = \begin{cases} 1 & if\ f(x) \leq T \\ 0 & if\ f(x) \geq T \end{cases} \qquad (2)$$

where T is a threshold($T \in R$), $f(x)$ is a mapping function, which will map x to a real number. The input of the algorithm is a set of similar and dissimilar pairs represented by GIST vectors. For the SSC training of SIFT feature, the input is a set of similar and dissimilar keypoints. In our training, similarity between images or keypoints is explicitly specified by their semantic meaning, and images of the same character or keypoints less than the threshold are regarded as similar training pairs. So it is very convenient to get the training set.

To get the threshold T in equation. 2, we need to traverse all the feasible thresholds in training pairs and select the one that maximize the TP−FP gap, where TP is the true positive rate and FP is the false positive rate classified by the threshold. For each dimension of x, set the minimum gap of TP−FP, we can

get some thresholds T and corresponding h functions. Two hash functions of GIST and SIFT feature we will get from the training, which can be used to map the GIST or SIFT vectors to binary codes.

This is the non-boost version of SSC (Shakhnarovich G, 2003), it trains very fast and works just well, but there are some limitations such as constrained geometry of projection and ignoring dependencies of dimensions. So we also use the boost version of SSC to gain better recognition result. Here is a brief overview of BoostSSC(Shakhnarovich G, 2003):

Each function h in equation. 1 naturally defines a classifier c: $x \to \{\pm 1\}$ on pairs of examples. To learn an ensemble classifier:

$$H(x) = sgn(\sum_{m=1}^{M} \alpha_m c_m(x)) \quad (3)$$

AdaBoost gives an iterative greedy algorithm that adds weak classifier c_m with an appropriate vote α_m one at a time. In our implementation, we set α_m to be a constant(0 or 1), so the distance between vectors can be evaluated by hamming distance. Throughout the iterations, AdaBoost maintains a distribution W which represents the weights of training pairs. In the training process, the distribution is updated so that the examples classified correctly have their weight reduced and those misclassified have their weight increased, here is the updating rule of W:

$$W_{m+1}(i) = W_m(i) exp(-\alpha_m l_m c_m(x_i))/Z_m^{AB} \quad (4)$$

where m is the iteration index, i is the example index, $l_m = \{\pm 1\}$ indicates the ground truth of similarity and c_m is the weak classifier which gives the prediction. Z_m^{AB} is a normalization constant given by:

$$Z_m^{AB} = \sum_{i=1}^{N} W_m(i) exp(-\alpha_m l_m c_m(x_i)) \quad (5)$$

at each iteration m, select the weak classifier c_m (h function) to minimize the training error:

$$r_m^{AB} = \sum_{i=1}^{N} W_m(i) l_i c_m(x_i) \quad (6)$$

After iteration, the hash functions will be obtained, which can be used to map input vector into binary codes.

At the training stage, the input of the SSC algorithm is a set of similar and dissimilar image pairs for GIST feature, in which similar and dissimilar keypoints are for SIFT feature. Similarity between images is explicitly specified by their semantic meaning. SIFT keypoints less than the threshold are regarded as similar pairs. After training, we can get GIST and SIFT hash functions, then we use the hash functions to convert feature library to binary codes, so the similarity of feature vectors can be measured by their hamming distances.

The size of each *GIST binary vector* is 216 bits, so it takes only 28 bytes. The GIST feature library has 41560 feature vectors, it will cost 1163680 bytes(1.11 MB). The length of the *SIFT binary vectors* in our implementation includes two parts: 4 bytes (one integer) used to store the number of SIFT vectors of each image; 128 bits(16 *bytes*) saves each single SIFT vector. For the SIFT feature library, it has 649219 feature vectors and 41560 integer numbers, which costs 10553744 bytes(10.06 *MB*), which is suitable for mobile platform. Table.1 shows the detailed information about the storage of GIST and filtered SIFT feature library before or after SSC encoding.

Table 1: The storage of GIST and SIFT features library before and after SSC coding.

	GIST		***SIFT***(*filtered*)	
	vector (byte)	*library (MB)*	*vector (byte)*	*library (MB)*
Before SSC Encoding	816	34	512	317.16
After SSC Encoding	28	1.1	16	10.06
Compressing Ratio	***96.73%***		***96.83%***	

3.4 Recognition

The recognition algorithm is the main part of our platform. To increase the recognition accuracy, we provide a candidate list of the most similar Chinese characters. In our method, we combine GIST and SIFT feature to ensure the recognition accuracy. The main idea is that we use SIFT feature to match these similar characters obtained by GIST feature and analyse the two matching procedure to give the final recognition result. So we first use GIST feature to find recognized image's $2k$ nearest neighbors (the same character has two features because of two typefaces) in an order as candidates. Then use SIFT feature to match them again and get the same Chinese characters in a new order by the number of feature points similar with the target image. Three points should be concerned here: **(a)** If the character in the first ten characters of the SIFT feature matching result is duplicate in all candidates. **(b)** If the distance of each character's location between GIST and SIFT matching result is more than ξ, where ξ is a constant. **(c)** The location of left Chinese characters in SIFT matching result will be as weight added to the GIST feature matching result and give them in a new order by weight. If a character

Table 2: Hardware configuration of the test device.

Items	Value
Phone	Samsung SM-P600
OPeration System	Android OS v4.4.2
CPU cores	4 cores + 4 cores
CPU Model	Exynos5420
CPU Frequency	1.3GHz+1.9GHz
GPU Model	ARM Mali-T628 MP6
Memory	16GB ROM + 3GB RAM
Camera Pixel	3264×2488

satisfies *(a)*, it will be added to the head vector. If a character satisfies *(b)*, it will be put into middle vector. The recognition result will consist of head vector, middle vector and *(c)*'s result. Finally, show the first k Chinese characters in that final ordered list as recognition result to users. A pseudocode description for the recognition process is given in Algorithm 2.

Algorithm 2: Recognize(L_1,L_2,*binGist*,*binSift*,k): recognize Chinese character images by GIST and SIFT features.

Input:
L_1 : The GIST feature binary library.
L_2 : The filtered SIFT feature binary library.
binGist : GIST binary vector of captured image.
binSift : SIFT binary vectors of captured image.
k : The number of characters as recognition result.
Output:
R : The string of recognition result.
1: Let $C = C_1, ..., C_n$ be the Chinese characters in L_1;
2: Let P_1 = {search the $2k$ most similar Chinese characters with *binGist* in L_1, the characters are in an order by the hamming-distance near *binGist*};
3: For each SIFT feature $Sift_i$ of C_i in P:
4: Let $MatchNum[i] = 0$;
5: For each v_ψ in $Sift_i$:
6: For each v_j in *binSift*:
7: $dist$= hamming-distance between v_j, v_ψ;
8: If $dist \leq threshold$:
9: $MatchNum[i] = MatchNum[i] + 1$;
10: Let P_2 = {$2k$ characters ordered by $MatchNum's$ value};
11: Let P_{head} = {the characters are duplicate and its location is in the first k characters of P_2};
12: Let P_{middle} = {the characters whose distance of its location between P_1 and P_2 is more than ξ };
13: Let P_{end} = {the left characters in an new order by combing each character's location in P_1 and P_2};
14: Let P_{final} = {the characters in an new order by combing P_{head},P_{middle} and P_{end}};
15: Let R = {the first k Chinese characters in P_{final}};

4 IMPLEMENTATION AND EXPERIMENT

4.1 Implementation

In this section, we will give the implementation of mobile Chinese character images recognition platform and the corresponding experiment results. As shown in Fig.1, there are two parts in the platform. The first part is implemented on PC, which includes creating the feature library and training the SSC hash functions. That stage is done off-line and need not be taken care of once finished. The second part is the Android application, which can recognize character images captured by phone camera. For GIST feature extraction, we use Java Native Interface (JNI) to implement the GIST extraction module on Android to make it faster. When the GIST vector is calculated, we use the hash functions obtained at the SSC training stage to encode the GIST vector to a binary vector.We use the library of OpenCV-2.4.9-andriod-sdk to extract SIFT feature on Android application, then use JNI to encode SIFT feature vectors to binary codes. Finally, we use JNI to recognize characters and return the result. Fig. 5 shows the screen capture of the Android application.

4.2 Experiment

We do some experiments on testing the recognition speed and accuracy of our platform on the test device(the hardware configuration is shown in Table.2).

4.2.1 Recognition Speed

Two procedures mainly affect the recognition speed: extracting GIST and SIFT features of a character image; searching binary vectors in the binary feature library. In the feature extracting, the pixels of images has great effect on the time consumption. We do some experiments on the feature extracting under different pixels shown in Fig. 6. When the image resolution is small as 32×32 pixels, the GIST extracting procedure costs only 78 milliseconds, and the feature matching time can be even shorter.

In the system, we scale the captured image to 32×32 pixels, because that image size is proper enough for the recognition, which has been proved by our experiments. It also reduces the computing complexity when extracting the GIST feature. The GIST feature library built on PC contains more than 40000 binary feature vectors. For SIFT feature extracting, we scale the captured image to 109×109 pixels, which is consistent with the size of character

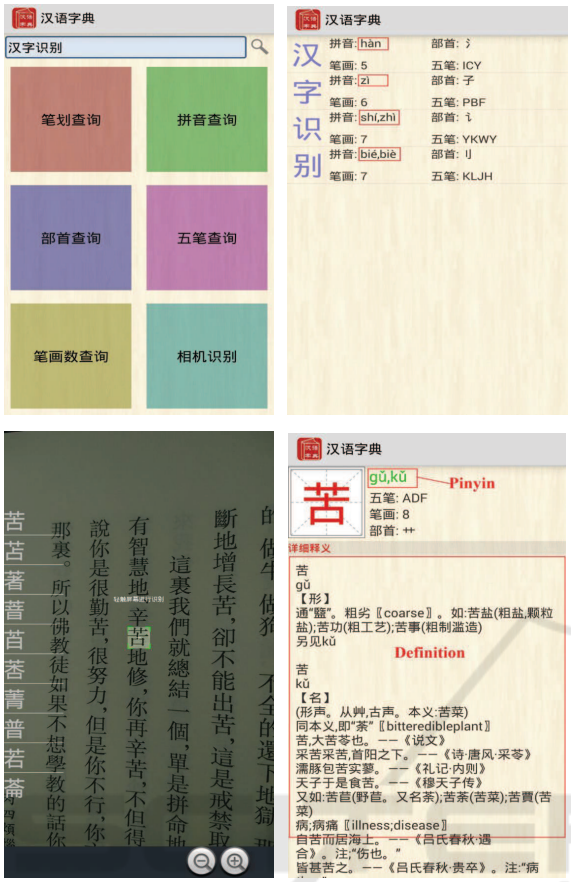

Figure 5: Screenshots of the Android recognition system.

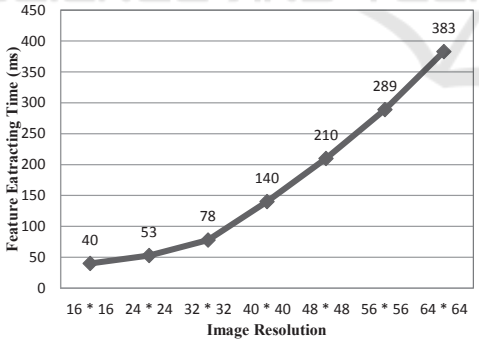

Figure 6: GIST extracting time on several image resolutions.

image in character images library. The time of SIFT extracting is much shorter than GIST's. Experimental results show that the total time for recognition is just about 200 milliseconds, which will deliver user a smooth experience.

4.2.2 Recognition Accuracy

For the Android apps of Chinese character images recognition, the most important requirement is the recognition accuracy. In order to increase the recognition ratio, we provide a candidate list that shows the most similar characters computed by our algorithm. Then users can choose the best one from these characters. Table.3 shows the recognition accuracy test for 400 Chinese characters randomly selected from Chinese books.

Table 3: Results for recognition accuracy test.

	Recognition Accuracy(%)
Top one	95
Top three	97.25
Top five	97.75
Top ten	98.25
Denial	**1.75**

We can see that most of the Chinese characters can be recognized and ranked top 10 in the candidate list. After recognition, users can select the right character and navigate to the paraphrase panel to see the detailed information.

In order to prove the efficiency of our recognizing algorithm, we also do some experiments on Chinese characters recognition compared to other Android apps. Although there exists many Android apps about Chinese character recognition, the accuracy is not very high. We installed Yun Mai, DaubNote, CamScanner, ABBYY TextGrabber four Android apps, which are famous and widely used on Android platform.

Four experiments are conducted to compare the recognition accuracy of our application with four popular Android apps. The first experiment is to recognize a short paragraph of the famous essay. Two experiments are to recognize the 400 most common simple Chinese characters. The fourth experiment is about the recognition of traditional characters. The samples of experiment are shown in Fig. 7. Table.4 shows the recognition accuracy for the experiment and proves that our application is much better than other four popular applications, especially for complicated characters.

Table 4: Comparison with popular recognition applications.

	a	*b*	*c*	*d*
Proposed	**100%**	**98%**	**98.5%**	**100%**
YunMai	96%	94.5%	78%	53.5%
DaubNote	**100%**	96.5%	94%	67%
CamScanner	**100%**	96.5%	93%	73.8%
TextGrabber	80%	82%	76%	43.5%

5 CONCLUSION

In this work, we present an algorithm using GIST and

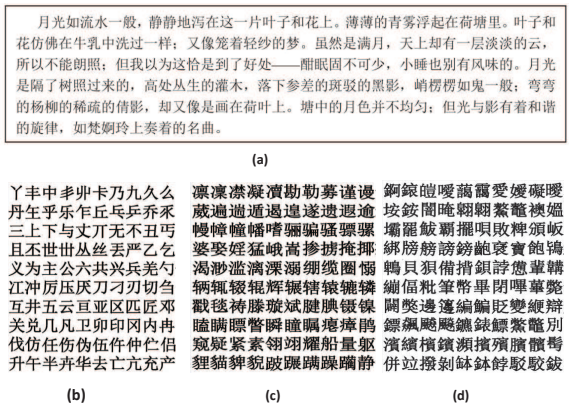

Figure 7: Samples of test set:*(a):short paragraph,(b):simple characters (stokes*< 10*),(c):simple characters (stokes*> 10*),(d):complicated characters*.

and SIFT feature to recognize Chinese character images. Before the recognition stage, we create the library by extracting feature from Windows installed font images. The methods of SIFT keypoints filtering and SSC encoding speed up the recognizing process and reduce the computational cost. Experiments show that the recognizer achieves great performance, but there is still space for improvement. Some complicate Chinese characters can't be recognized correctly due to the instability of the phone camera and limited feature library. In the future, we will improve the application by applying some image preprocessing method to alleviate the influence of hands trembling and adding more Chinese character feature vector to the feature library.

ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China(No.61379073) and the CADAL Project and Research Center, Zhejiang University. Thank all the reviewers for helping us to improve our work.

REFERENCES

Barbu, T. (2009). Content-based image retrieval using gabor filtering. In *Database and Expert Systems Application, 2009. DEXA'09. 20th International Workshop on*, pages 236–240. IEEE.

Calonder M, Lepetit V, S. C. (2010). Brief: Binary robust independent elementary features. In *Computer VisionCECCV 2010. Springer Berlin Heidelberg*, pages 778–792. Springer.

Jin, Z., Qi, K., Zhou, Y., Chen, K., Chen, J., and Guan, H. (2009). Ssift: An improved sift descriptor for chinese character recognition in complex images. In *Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on*, pages 1–5. IEEE.

Lin, Y., Wu, J., Gao, P., Xia, Y., and Mao, T. (2013). Lsh-based large scale chinese calligraphic character recognition. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pages 323–330. ACM.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Mikolajczyk K, S. C. (2005). A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630.

Ni, K., Kannan, A., Criminisi, A., and Winn, J. (2008). Epitomic location recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175.

Pengcheng, G., Jiangqin, W., Yuan, L., Yang, X., and Tianjiao, M. (2014). Fast chinese calligraphic character recognition with large-scale data. *Multimedia Tools and Applications*, pages 1–18.

Rosenberg, A. and Dershowitz, N. (2012). *Using SIFT Descriptors for OCR of Printed Arabic*. PhD thesis, Citeseer.

Shakhnarovich G, Viola P, D. T. (2003). Fast pose estimation with parameter-sensitive hashing. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 750–757. IEEE.

Tianjiao, M., Jiangqin, W., Pengcheng, G., Yang, X., and Yuan, L. (2013). Calligraphy word style recognition by knn based feature library filtering. In *3rd International Conference on Multimedia Technology (ICMT-13)*. Atlantis Press.

Zhang Z, Jin L, D. K. (2009). Character-sift: a novel feature for offline handwritten chinese character recognition. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 763–767. IEEE.

Zhen-Yan, W. (2014). Chinese character recognition method based on image processing and hidden markov model. In *Intelligent Systems Design and Engineering Applications (ISDEA), 2014 Fifth International Conference on*, pages 276–279. IEEE.

Zhuang, Y., Zhang, X., Lu, W., and Wu, F. (2005). Web-based chinese calligraphy retrieval and learning system. In *Advances in Web-Based Learning–ICWL 2005*, pages 186–196. Springer.