Pattern Analysis and Applications
https://doi.org/10.1007/s10044-018-00776-x

THEORETICAL ADVANCES

A novel nearest interest point classifier for offline Tamil handwritten character recognition

R. N. Ashlin Deepa[1] · R. Rajeswara Rao[2]

Received: 13 June 2018 / Accepted: 26 December 2018

Abstract
Handwritten character recognition is the most widely used branch of study in image pattern recognition. Tamil, the official language of Tamil Nadu in South India, Sri Lanka, Singapore and Malaysia, has a script which contains many loops and compound characters, with small differences between character classes. Most of the research on offline Tamil handwritten character recognition system was done only on few character classes as it is very difficult to distinguish between minute dissimilarities of large character classes. It is important to design a complete recognition system that can process all character classes of Tamil and distinguish natural variability between inter-class images. Unlike conventional machine learning approaches for pattern recognition problems, we have proposed a nearest interest point classifier, which can choose sufficient and necessary subset of features from a variable length high dimensional feature vector. Since this is a practical problem, in this work, a study on image to image matching is included through feature analysis without using machine learning approaches. The proposed algorithm gave a good recognition accuracy for all the character classes on the standard database available for Tamil, HP Labs offline Tamil handwritten character database. Our proposed classifier produced a recognition accuracy of 90.2% while including the whole dataset. The method has been compared with the standard classifiers and has been proved to be a state-of-the-art performance in recognition of accuracy over the previous results given in the literature.

Keywords Handwritten character recognition · Pattern recognition · Classification · Variable length · High dimensional data · Speeded up robust features

1 Introduction

India is a multilingual country where people in different states speak different languages. Research says our country is multiscript, with 18 languages and 12 different major scripts [1]. Indians use their own native language as a media of communication. Keyboard is the interface between human and machine for communication. To make this more effective and easy, *handwritten character recognition* (*HCR*) systems are given high importance, which accepts handwritten documents as input and converts it into machine readable form. The popular applications of handwritten character recognition system include (1) reading aid for the blind, (2) automatic text entry into the computer for desktop publication, library cataloging, ledgering, etc. (3) automatic reading for sorting of postal mail, bank checks and other documents and (4) language processing [2]. Basically, handwritten character recognition system is classified into two, based on the form in which the information is fed to the system: *online and offline*. In *offline* systems, the information is written on a paper and digitized by a scanner and presented to the computer as an image. In contrast, in *online* systems, the user uses a stylus to write the input through a digitizing tablet which captures temporal information such as (x, y) coordinates at evenly spaced time intervals [3, 4]. Offline systems are more complex than online systems. HCR systems are widely available for foreign languages English, Roman, Japanese, Chinese, Korean and Arabic scripts [5–8]. But, very little steps are taken to develop a system for Indian languages, especially for South Indian Languages. Our work

✉ R. N. Ashlin Deepa
deepa.ashlin@gmail.com

R. Rajeswara Rao
raob4u@yahoo.com

1 Gokaraju Rangaraju Institute of Engineering and Technology, Bachupally, Kukatpally, Hyderabad, Telangana, India

2 JNTUK University College of Engineering, Vizianagaram, Andhra Pradesh, India

presents an offline HCR system for the oldest South Indian script, 'Tamil.'

1.1 Tamil

Tamil is a popular South Indian script with millions of speakers within India and abroad, and is the administrative language of Tamil Nadu, a Southern state in India. The people in urban area are familiar with *English* and information technology for communication. But, the rural people depend on the native language for reading and writing. Therefore, providing interaction with computer in their native language and in a natural way such as handwriting is absolutely necessary [9]. Tamil is a member of Dravidian language family, and its script is said to be developed from *'Brahmi script.'* It is written in a left-to-right fashion. Although it has been influenced by Sanskrit to a certain degree, Tamil along with other South Indian languages are genetically unrelated to the descendants of Sanskrit such as Hindi, Bengali and Gujarati. Most Tamil letters have circular shapes; partially due to the fact that they were originally carved with needles on palm leaves, a technology that favoured rounded shapes [10]. Tamil is the first language to be conferred upon as a classical language by the government of India in 2004, followed by Sanskrit [11, 12]. The Tamil script contains twelve *vowels*, eighteen *consonants* and one *aytham*. The vowels in combination with consonants form 216 composite consonants and in total of 247 different characters. The script also includes five *granthas* originated from *Sanskrit.* Since the consonant is represented as separate character while forming a composite consonant, it can be broken into separate symbols. Thus, the total number of characters to be recognized is counted as 156.

1.2 hpl-tamil-iso-char database

The non-availability of benchmark database is the major obstacle in HCR of Tamil. Recently, HP Labs India has established a database called *'hpl-tamil-iso-char'* [13], with 156 different character classes which can be freely downloaded from the website.[1] Figure 1 gives 156 characters of the Tamil dataset used in recognition process. The database contains approximately 500 samples in character classes. The samples were collected from native Tamil writers including adults, university graduates and school children from different cities of India.

[1] Lipi Toolkit from HP Labs is an open-source tool for data collection and is freely available for download at http://lipitk.sourceforge.net.

1.3 Related work

An efficient HCR system is not available for South Indian scripts, specifically for Tamil scripts [14, 15]. Few researches already exist in *online* character recognition system for Tamil script with better recognition accuracies because of the temporal information such as (x, y) coordinates, captured on the digitizing tablet. Connell and Jain [16] proposed template-based method where templates can be viewed as representing different styles of writing a particular character. These templates are then used as a reference for efficient classification using decision trees, and 86.9% accuracy was achieved for 36 character classes. Kunwar and Ramakrishnan [17] used fractal codes as features and DTW for distortion evaluation during classification and encoding processes and scored 90% accuracy for 20 training samples and 50 testing samples from HP Labs database. Sundaresan and Keerthi [18] proposed an algorithm based on sequence of cosine of angle along with wavelet features and used neural network for classification. The recognition accuracy is 96.54% for 12 character classes with 50 samples for training and 200 samples for testing. This method also produced 94.30% for 135 character classes with 40 samples per class for training and 20 samples for testing. We can observe that the recognition accuracy decreases when the number of classes under consideration increases. Aparna et al. [19] defined a structure- or shape-based representation of a stroke where a stroke is represented as a string of shape features. Using this string representation, an unknown stroke is identified by comparing it with a database of strokes using a flexible string matching procedure.

Coming to offline HCR, few works can be noted with good recognition accuracy when less number of classes used for training and testing. In an approach of topological matching [20], correlation coefficients were used as feature vectors from a manually collected database of 36 classes to produce a recognition accuracy of 60%. Fuzzy concept was used over the distance from the frame along with a suitable membership function to produce an average accuracy of 90%, and the samples are collected manually and used for seven classes of 200 samples [21]. In another approach [22], a statistical classifier based on interval estimation was used with pixel density as feature from 26 classes to produce an accuracy of 79.9%, on 1000 and 800 samples for training and testing, respectively. Backpropagation NN was used with Fourier descriptors as features over 30 character classes, and 97% accuracy was produced as the highest score in offline recognition [23]. But, to be noted, the number of character classes used in the experiment was 30. In [24], Hu's invariant moments and Zernike moments are used as features and feedforward neural network was used for classification of 36 character classes with 100 samples. HMM was applied on a sequence of straight lines extracted from character images, and an average accuracy of 94% was produced for

Fig. 1 One hundred and fifty-six Tamil character classes from the HP dataset [13]

அ	த	ஙி	ஷி	ழீ	மு	மூ	க்ஷு	ல்
ஆ	ந	சி	ஜி	ளீ	யு	யூ	ஸூ	வ்
இ	ப	ஞி	ஹி	றீ	ரு	ரூ	ஹூ	ழ்
ஈ	ம	டி	க்ஷி	னீ	லு	லூ	ஜூ	ள்
உ	ய	ணி	கீ	ஸீ	வு	வூ	ஹூ	ற்
ஊ	ர	தி	ஙீ	ஷீ	ழு	ழூ	க்ஷூ	ன்
எ	ல	நி	சீ	ஜீ	ளு	ளூ	க்	ஸ்
ஏ	வ	பி	ஞீ	ஹீ	று	றூ	ங்	ஷ்
ஐ	ழ	மி	டீ	க்ஷீ	னு	னூ	ச்	ஜ்
ஒ	ள	யி	ணீ	கு	கூ	◌ா	ஞ்	ஹ்
ஓ	ற	ரி	தீ	ஙு	ஙூ	ெ◌	ட்	க்ஷ்
ஃ	ன	லி	நீ	சு	சூ	ே◌	ண்	ஔ
க	ஸ	வி	பீ	ஞு	ஞூ	ை◌	த்	
ங	ஷ	ழி	மீ	டு	டூ	ஸ்ரீ	ந்	
ச	ஜ	ளி	யீ	ணு	ணூ	ஸு	ப்	
ஞ	ஹ	றி	ரீ	து	தூ	ஷு	ம்	
ட	க்ஷ	னி	லீ	நு	நூ	ஜு	ய்	
ண	கி	ஸி	வீ	பு	பூ	ஹு	ர்	

word recognition [25]. Pal et al. [26] developed a system with 64-dimensional feature for high-speed recognition and a 400-dimensional feature for high accuracy recognition and used a quadratic classifier and tested for 36 character classes of 10,216 samples to produce a recognition accuracy of 96.73%. All the above-mentioned methods in the literature were experimented on few character classes of manually collected samples and produced fair recognition accuracies.

The literature also shows that when the number of character classes under experiment increases, the recognition accuracy decreases. Pixel density features were used with SVM classifier and produced 82.04% accuracy for 106 classes which include vowels, consonants and all composite consonants [27]. The data were collected manually from different users, and 35,441 training samples and 6048 test samples were used. The only standard database available for offline Tamil character recognition is developed by HP Labs. We have used this database for our previous works. To the best of our knowledge, apart from us, only two studies on offline Tamil character recognition gave the result by using this database. Vijayaragavan et al. [28] used HP Labs database and produced 94.4% recognition accuracy on 35 character classes by using features of stochastic pooling, probabilistic weighted pooling and local contrast normalization, and convolutional neural network as classifier. Bhattacharya et al. [29] used HP Labs database for offline character recognition system for Tamil, including all samples of 156 character classes, and an accuracy of 89.6% was produced. In our previous work [30], Zernike moments and geometrical features were given to neural network and an accuracy of 93.4% was obtained for nine classes with nine samples each, from HP Labs database. In another work [31] of 10 character classes with 150 training samples and 50 test samples, we obtained an accuracy of 78.97%. In our last method, *'Modified GA'* [32], line strokes were used as features and genetic algorithm was used for classification. We have used HP Labs database with all 156 character classes, but, only 3900 training samples and 3120 test samples produced an accuracy rate of 89.5%. Table 1 shows the details of literature in offline Tamil HCR and the impact of number of character classes used and size of dataset on the recognition accuracy.

The above literature shows the decline in recognition accuracy when the number of character classes and test samples increases. This is because of the availability of similar-shaped character classes in Tamil script, and finding a difference among those character classes is a challenge for the recognition system. Minute parts make a distinction between character classes, and a small variation in the size and shape of the character image makes a huge impact on extracted features. Tamil characters naturally contain complex structures with enormous curves and loops and thus making feature extraction a tough task. So, *three* points are considered in our work for the selection of feature extraction and classification methods. *First* a feature extraction technique, which analyzes the physical structure of individual character irrespective of its size and shape, is useful in this scenario. *Second*, an approach to increase inter-class variability may be adapted. *Third,* a simple classification method can be developed, in order to avoid the computational complexities of machine learning approaches but not compromising with performance. These three points are analyzed, and appropriate methods are developed to improve recognition accuracy of offline Tamil handwritten characters.

Many feature extraction methods have been discussed in [33–36]. In our work, *speeded up robust features* (*SURF*) [37], a scale, rotation and translation invariant feature descriptor, is considered. SURF descriptor is already used in many classification problems [38–40]. SURF generates variable number of *interest points* (*IP*) from meaningful physical structure of the image. The number of IPs generated varies from image to image. In order to increase the inter-class variability, a threshold approach on each IP of the training image is adapted which differentiates the inter-class variability between training image and test image. Each IP of SURF contains 64-dimensional feature vector. The number of IPs generated varies from image to image. As an average, 80–120 IPs are generated from every image. So, if an image contains 80 IPs as features, the classification algorithm should be able to handle feature vector of 80×64. In such cases, HCR becomes a very complex problem with high dimensional data and natural variability.

A significant *increase in feature dimensionality* complicates the classifier performance because the convergence to a classification algorithm is slow and incorrect in this case. When a good feature vector is fed through an appropriate classifier algorithm, the performance of the recognition process increases. *Variable length* feature vector is another problem for most of the classifiers as they are defined to use fixed length strings. Few approaches are already used in order to handle variable length data [32, 41, 42]. But, these approaches increase the complexity of data storage and computational complexity.

A wide range of classification problems have been solved by simple algorithms using *nearest neighbor* (*NN*) principle [43–47]. The main idea of NN principle is to calculate the global distances between the comparing patterns followed by ranking to decide the NN that best determines the class of a test pattern. Increase in feature dimensionality produces very slow and inaccurate convergence to a classification problem [48, 49]. In order to address these problems, the conventional solution is to depend on feature extraction and feature selection methods [50, 51]. The use of machine learning techniques reduces the inaccuracies that arise due to distance calculations [52–54]. But, these methods suffer from optimization problems that require feature dimensionality reduction and computational complexity. Calculation of distance fails completely in high dimensional database when it is often difficult to trace redundancy of data. A modification to NN classifier is proposed to reduce complexities, and the decision on class of test data is taken by selecting *nearest features* (*NF*), based on threshold, from the training data [55]. But, NF classifier deals with fixed length feature vector. Performance improvement in NN has been tried in many ways: assigning unique weights to each NN [56], choosing subset of weighting features according to their discriminatory power [57], generalizing the basic k-NN classifier and introducing feature voting that considers each feature separately [58, 59]. Since NN classifier is a basic classifier and is popular for its simplicity, adding complexity to improve its performance negates the comfort of using it for the classification problems.

If a simple classifier is used along with feature set produced from SURF descriptors, the computation complexities of standard machine learning approaches can be avoided. As different from normal and complicated machine learning algorithms, we propose a simple classifier, variation of NN, called *nearest interest point* (*NIP*) *classifier*, which is applicable to real-time applications such as HCR that gives robust classification performance. The novelty of our work is given in highlights.

1.3.1 Highlights

a. Proposal of NIP classifier to maintain simplicity in classification

b. Variable length and high dimensional feature vector is handled easily

Table 1 Summary of the literature on Tamil handwritten character recognition and related results showing the number of classes and samples used for experiment

References	Features	Classifier used	Dataset used		Recognition accuracy
			Name	Size	
Suresh et al. [21]	Distance from the frame and a suitable membership function	Fuzzy concept	Manually collected	Classes: 7 200 samples	90% (average % of accuracy calculated from paper)
Wahi et al. [24]	Hu's invariant moments and Zernike moments	Feedforward neural network	Manually collected	Classes: 36 100 samples Training: 75% of data Testing: 25% of data	Not given
Jagadeesh Kumar et al. [25]	Sequence of straight lines	HMM	Manually collected handwritten documents	Not mentioned	Count of words (accuracy %) 25 (96.4) 50 (95.6) 100 (94.8) 150 (93.7) 200 (93.0)
Hewavitharana and Fernand [22]	Pixel density	Statistical classifier based on interval estimation	Manually collected	Classes: 26; Training: 1000 Testing: 800	79.9%
Chinnuswamy and Krishnamoorthy [20]	Correlation coefficients	Topological matching	Manually collected	Classes: 36	60%
Vijayaraghavan et al. [28]	Features by using stochastic pooling, probabilistic weighted pooling and local contrast normalization	ConvNets	*HP labs hpl-tamil-iso-char*	Classes: 35, Samples: 18,535	94.4%
Pal et al. [26]	64-dimensional feature for high-speed recognition and a 400-dimensional feature for high accuracy recognition	Quadratic classifier	Manually collected	Classes: 36 Samples: 10,216	96.73%
Sutha and RamaRaj [23]	Fourier descriptors	Backpropagation NN	Manually collected	Classes: 30	97%
Shanthi and Duraiswami [27]	Pixel density	SVM	Manually collected from 117 different users	Training: 106 classes (35,441 samples) Testing: 34 classes (6048 samples)	82.04%
Bhattacharya et al. [29]	Chain code histogram features	k-means clustering and MLP	*HP labs hpl-tamil-iso-char*	Classes: 156	*89.66%*
Ashlin Deepa and Rajeswara Rao [31]	Eigenvalues	PCA	***HP labs hpl-tamil-iso-char***	Classes: 10 Samples: 200 Training: 150 Testing: 50	78.97%
Ashlin Deepa and Rajeswara Rao [30]	Zernike moments and diagonal-based features	Neural network	***HP labs hpl-tamil-iso-char***	Classes: 9 Samples: 9	93.8%
Ashlin Deepa and Rajeswara Rao [32]	Horizontal, vertical, right-slanted and left-slanted strokes	GA	***HP labs hpl-tamil-iso-char***	Classes: 156 Training: 3900 (25 samples per class) Testing: 3120 (20 samples per class)	89.5%

c. Inter-class dissimilarity is opened up by providing proper thresholding approach

d. Overall computational complexity has been reduced because of the simplicity of our proposed method

2 Nearest interest points (NIPs) from SURF descriptors

The main concept of proposed method is making local-level decision on the class of the test image based on individual features called *IPs* of the training images. The global decision on the class of the test image is obtained after processing these local decisions. To explain NIP, let us consider a simple example from Table 2, with training set: $T\{C_{11}, C_{21}, C_{31}, C_{12}, C_{22}, C_{32}, C_{13}, C_{23}\}$. Let C_{jc} be the *j*th image of the class *c*. The training set consists of images from three classes (1) C_1 (2) C_2 and (3) C_3. The number of SURF IPs generated varies from image to image. Let In_{ijc} be the *i*thIP of *j*th image from class *c*. Each IP In_{ijc} of training image C_{jc} is associated with a threshold value θ_{ijc}. Each IP is characterized by a fixed dimension (64) descriptor vector. A comparison between two character images is performed by calculating *M InterestPoint-to-InterestPoint* (*IP-to-IP*) distances between IPs of the images C_{jc} *and Z* where Z is the test image and C_{jc} is having *M* IPs. Then, the *IPs* of C_{jc} having distances smaller than the threshold distance θ_{ijc} are defined as *nearest interest points* (*NIP*) of C_{jc} for the test image *Z*. The role of the parameter *k* in *k-NN* at the global level is played by the threshold θ_{ijc} at feature level. This means there are as many θ_{ijc} values to be enhanced as the number of *IPs*.

In online applications such as anti-phishing system, single threshold approach works well as it is more concerned about timely detection of hackers [60]. Hence, adding more threshold values will allow more malfunctioning of many applications. But, our application aims at developing an offline HCR system, where accuracy is concerned more than reducing the time of classification. Though the value of θ_{ijc} varies from *IP-to-IP*, this method helps in using only few *IPs* with small distance values for classification, instead of using all IPs extracted by SURF. To implement this, the statistics of the distances of *i*th *IP* (In_{ijc}) of the image C_{jc}, with inter-class training images, are used to calculate the threshold value θ_{ijc} of In_{ijc}. The unique threshold value is set as $\theta_{ijc}=2\sigma_{ijc}$, where σ_{ijc} is the standard deviation of the inter-class IP distances of In_{ijc} of C_{jc}. The number of *NIPs* of each training image C_{jc} is used to calculate *image similarity scor*e which is used in the global decision on the class of the test image.

We will use the simple example with few *IPs,* to introduce the steps needed to calculate *NIP vote*s and *image similarity score for global decision*. In general, the number of IPs generated by SURF will be more than 70 for most of the images. As it is not possible to give all IPs and show the distance calculation, an example dataset T is taken with three character classes and eight training images in Table 2. The images in the training set are given maximum of 5 IPs. The objective is to identify the unknown class label of the test image *Z*, by comparing it with a set of images in training set *T*. It is shown in Table 2 that the training set is formed of three images from class C_1, three images from class C_2 and two images from class C_3. Note that this database is not real which is used for experiment. It is presented only in paper to illustrate the concept of *NIP classifier.* The actual experiment was done on HP database for Tamil characters.

2.1 Determination of distance between IP-to-IPs

Consider the training set *T*, in which *IPs* of an image *j* having a class label *c* be $\left\{G = \left\{In_{pj_c}, \quad \forall p,j,c \in [1,X], \quad X \in Integer\right\}\right.$ and IPs of an image *n* having a class label $\bar{c}$ be $\left\{H = \left\{In_{qn_{\bar{c}}}, \quad \forall q,n,\bar{c} \in [1,X], \quad X \in Integer\right\}\right.$ where $c \neq \bar{c}$ and $c, \bar{c}$ ϵ class label and *p and q* varies from 1 to number of IPs of images *j* and *n,* respectively. Let us calculate the distance between two images *j* and *n* by calculating the distance between their IPs. In_{ij_c} *is i*th IP of image *j* from class *c* and $In_{mn_{\bar{c}}}$ is *m*th IP of image *n* from class $\bar{c}$. The distance between IPs In_{ij_c} and $In_{mn_{\bar{c}}}$ can be calculated using a simple trigonometry as shown in Fig. 2. The distance of the *IPs* from the center line may be considered. These IPs would take the same values at any point in the line following RO. Then, we have $|In_{ij_c}/In_{mn_{\bar{c}}}| = 1$ or $\left|In_{ij_c} - In_{mn_{\bar{c}}}\right| = 0$. However, the point F will have $|In_{ij_c}/In_{mn_{\bar{c}}}| > 1 or \left|In_{ij_c} - In_{mn_{\bar{c}}}\right| > 0$. The distance between the points R and F is denoted as $d'\left(= \left|In_{ij_c} - In_{mn_{\bar{c}}}\right|\right)$, while that between O and H is denoted as *d*. Since the slope of the line RO is always 1,$\angle ORF = 45°$, and as the line OF is perpendicular to the x-axis, $\angle OFR = 90°$ and since the acute angles of a right triangle are complementary, $\angle ROF = 45°$. Further, as the lines RO and FH are parallel to each other, $\angle HOR = 90°$, which forces $\angle ROF = \angle FOH = \angle OFH = 45°$. This means, ΔOHF *to be isosceles* and so the length of OH to be equal to length of FH represented as *d*. Applying Pythagoras theorem on $\Delta OFH, d'^2 = d^2 + d^2$ or $d = d'/\sqrt{2}$ and can be represented as

$$d = \frac{\left|In_{ij_c} - In_{mn_{\bar{c}}}\right|}{\sqrt{2}} \tag{1}$$

The distance between two IPs In_{ij_c} and $In_{mn_{\bar{c}}}$ can be found using (1). This process is repeated for all the IPs of *n* in *H,* and the minimum distance is found using

$$di = \min\left\{\frac{\left|In_{ij_c} - In_{qn_{\bar{c}}}\right|}{\sqrt{2}}\right\}, \tag{2}$$

Table 2 NIP vote calculation to identify the class of unknown object Z from C1, C2 and C3 classes

Image similarity score for NIPs
IPs of the test image Z are InZ_1, InZ_2, InZ_3, InZ_4 and InZ_5

Class label	Object label	IPs of training images					IP-to-IP distances between IPs of training images and test image Z					NIP votes, *Vi*						Image similarity score	
							d_1	d_2	d_3	d_4	d_5							Total NIP votes *vi*	Normalized image similarity score
C1	C_{11}	In_{111}	In_{211}	In_{311}	In_{411}		($In_{111,}$ InZ_1)	($In_{211,}$ InZ_3)	($In_{311,}$ InZ_5)	($In_{411,}$ InZ_4)		Threshold	θ_{111}	θ_{211}	θ_{311}	θ_{411}		2	2/4 = 0.5
							0.1069	0.236	0.0083	0.317		Values	0.31	0.1082	0.0932	0.1542			
												Vi	1	0	1	0			
	C_{21}	In_{121}	In_{221}	In_{321}	In_{421}	In_{521}	($In_{121,}$ InZ_2)	($In_{221,}$ InZ_1)	($In_{321,}$ InZ_3)	($In_{421,}$ InZ_4)	($In_{521,}$ InZ_5)	Threshold	θ_{121}	Θ_{221}	Θ_{321}	Θ_{421}	Θ_{521}	**5**	**5/5 = 1**
							0.0913	0.0095	0.0402	0.0762	0.0081	Values	0.3123	0.1487	0.3267	0.0928	0.2154		
												Vi	1	1	1	1	1		
	C_{31}	In_{131}	In_{231}	In_{331}	In_{431}		($In_{131,}$ InZ_1)	($In_{231,}$ InZ_2)	($In_{331,}$ InZ_3)	($In_{431,}$ InZ_5)		Threshold	θ_{131}	θ_{231}	θ_{331}	Θ_{431}		3	3/4 = 0.75
							0.4302	0.0921	0.0761	0.0091		Values	0.1814	0.2217	0.3012	0.1325			
												Vi	0	1	1	1			
C2	C_{12}	In_{112}	In_{212}	In_{312}	In_{412}		($In_{112,}$ InZ_1)	($In_{212,}$ InZ_4)	($In_{312,}$ InZ_3)	($In_{412,}$ InZ_5)		Threshold	Θ_{112}	θ_{212}	Θ_{312}	Θ_{412}		1	1/4 = 0.25
							0.8024	0.3542	0.2831	0.6767		Values	0.3853	0.1479	0.4092	0.2952			
												Vi	0	0	1	0			
	C_{22}	In_{122}	In_{222}	In_{322}	In_{422}	In_{522}	($In_{122,}$ InZ_1)	($In_{222,}$ InZ_3)	($In_{322,}$ InZ_5)	($In_{422,}$ InZ_2)	($In_{522,}$ InZ_4)	Threshold	Θ_{122}	θ_{222}	θ_{322}	θ_{422}	θ_{522}	0	0
							0.4884	0.4376	0.7503	0.8313	0.5782	Values	0.2165	0.1983	0.3498	0.3269	0.2948		
												Vi	0	0	0	0	0		
	C_{32}	In132	In232	In332	In432	In532	($In_{132,}$ InZ_2)	($In_{232,}$ InZ_1)	($In_{332,}$ InZ_4)	($In_{432,}$ InZ_3)	($In_{532,}$ InZ_5)	Threshold	θ_{132}	θ_{232}	Θ_{332}	θ_{432}	θ_{532}	1	1/5 = 0.2
							0.2962	0.6712	0.506	0.551	0.2154	Values	0.312	0.3574	0.4187	0.2476	0.1962		
												Vi	1	0	0	0	0		
C3	C_{13}	In113	In213	In313			($In_{113,}$ InZ_2)	($In_{213,}$ InZ_3)	($In_{313,}$ InZ_1)			Threshold	θ_{113}	θ_{213}	θ_{313}			0	0
							0.5784	0.6053	0.7902			Values	0.2683	0.29	0.3782				
												Vi	0	0	0				
	C_{23}	In123	In223	In323	In423	In523	($In_{123,}$ InZ_1)	($In_{223,}$ InZ_3)	($In_{323,}$ InZ_2)	($In_{423,}$ InZ_5)	($In_{523,}$ InZ_4)	Threshold	Θ_{123}	θ_{223}	θ_{323}	θ_{423}	Θ_{523}	3	3/5 = 0.6
							0.529	0.0927	0.0285	0.6277	0.0154	Values	0.2285	0.4332	0.3901	0.2815	0.1962		
												Vi	0	1	1	0	1		

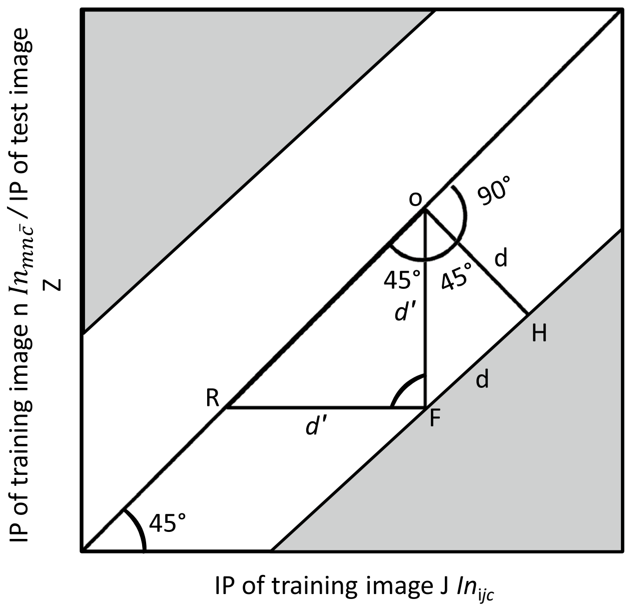

o
90°
d
45°
45°
d'
H
d
R
F
d'
45°
IP of training image n $In_{mn\bar{c}}$ / IP of test image Z
IP of training image J In_{ijc}

Fig. 2 Graphical representation for the calculation of the distance d using simple trigonometric relationships. The value of d is proportional to σ, where σ is the net standard deviation of inter-class IP differences between training images

where q varies from 1 to number of IPs in image n. The mth *IP of n* $In_{mn_{\bar{c}}}$, with minimum distance *di,* is taken to form a matching pair ($In_{ij_c}, In_{mn_{\bar{c}}}$). The meaning is ith IP of image j from class c formed minimum distance with mth IP of image n from class $\bar{c}$. This process is repeated for all the IPs of set G to find matching pairs of *IPs* of image j with image n.

An example from Table 2 is given in Fig. 3, where C_{11}, the first image of class *C1,* generated 4 IPs ($In_{111}, In_{211}, In_{311}$ and In_{411})) and image Z generated 5 IPs (*InZ_1, InZ_2, InZ_3, InZ_4 and InZ_5*). (The number of IPs is reduced for the purpose of explanation.) Let In_{111} be the *first* IP of first image (C_{11}) of class *C1.* The distance between In_{111} and each of the 5 IPs of image Z is calculated. It is found that In_{Z_1}, the first IP of image Z, produced minimum distance with In_{111} to form a matching pair (In_{111},In_{Z-1}). This process is repeated for all the IPs of the image C_{11} to produce the matching pair with the IPs of the image Z. Only four matching pairs with smaller distances are considered in the given example, as shown in Fig. 3.

2.2 Calculation of NIP threshold value θ_{ijc}

The idea of NIPs is implemented as a conversion of the IP-to-IP distances (d_i) to NIP local decisions labeled as *NIP votes* (*vi*) in Table 2. Each IP of the training image is associated with a NIP threshold value θ_{ijc} which is obtained from the standard deviation of inter-class IP-to-IP distances (the distance is calculated between the descriptor vectors of the IPs) of the images. The process of distance calculation between IP In_{ij_c} of image j, and image n and following matching pair formation ($In_{ij_c}, In_{mn_{\bar{c}}}$) is explained in Sect. 2.1. In order to calculate threshold value θ_{ijc} for In_{ij_c} of image *j of class c*, a set of distance values $\{di\}$ of In_{ij_c} with all the inter-class-images in the training set are calculated and matching pairs are formed. This set of calculated distances $\{di\}$ give inter-class IP-to-IP distances of In_{ij_c}. The standard deviation σ_{ijc} of distance values *{di}* is calculated which gives threshold value θ_{ijc}= 2 σ_{ijc}, of In_{ij_c}. This process is repeated for each IP of training image j to calculate their threshold values.

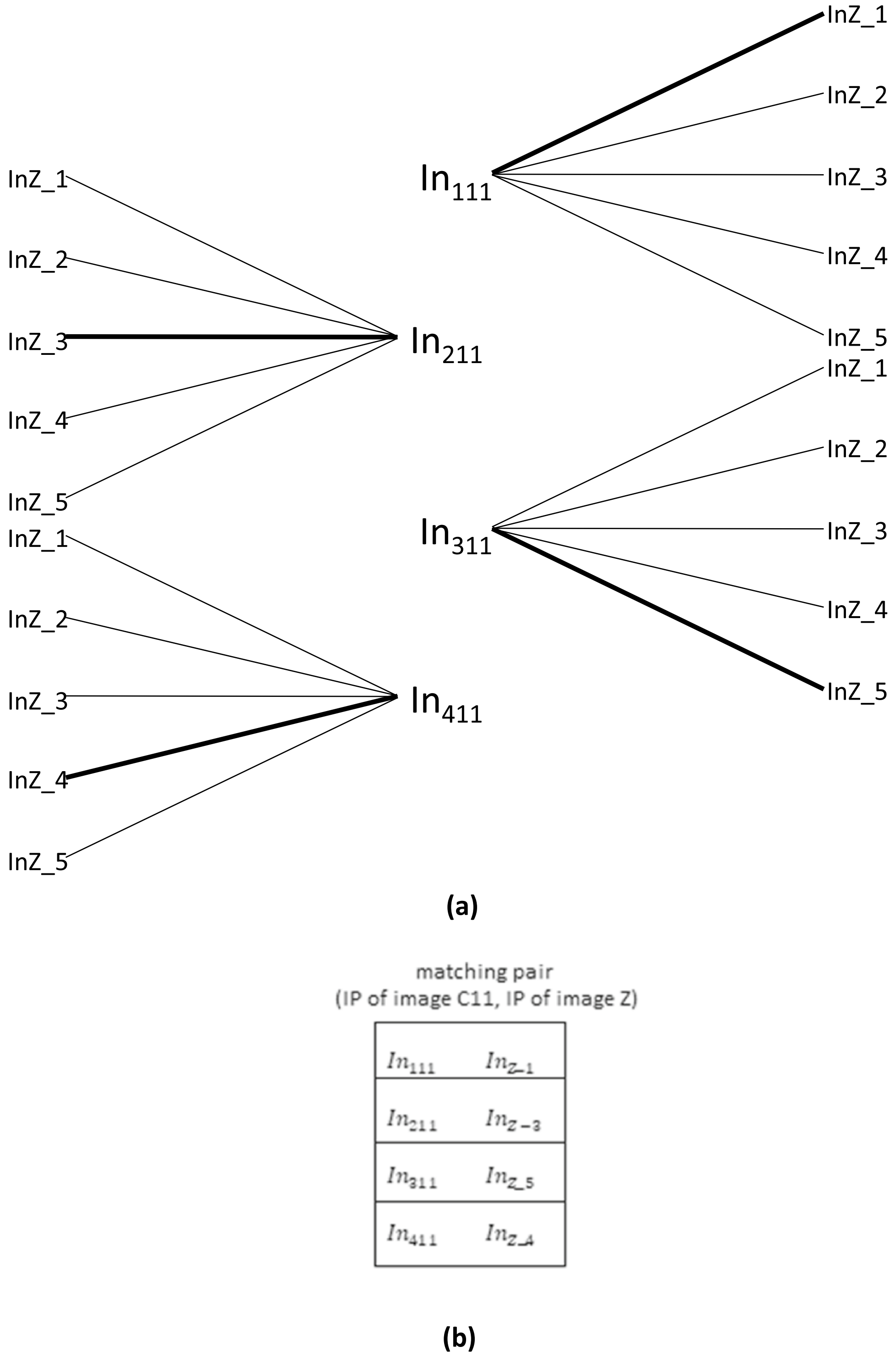
The use of the training set in Table 2 results in a total of 5 *di* values (inter-class IP-to-IP distances) for the IP In_{111} of C_{11}. These 5 *di* values comprise of three inter-class distances between C_{11} and images from class C_2 {C_{12}, C_{22} and C_{32}} and two inter-class distances between C_{11} and images from class C_3 {C_{13} and C_{23}}. The standard deviation of the inter-class *IP-to-IP* distance, σ_{111}, represents the relative degree of closeness of IP In_{111} of C_{11} with different images from class C_2 *{C_{21}, C_{22} and C_{23}}* and from class C_3 {C_{31} and C_{32}}.

Definition 1 The standard deviation $\sigma_{ijc} = \sqrt{\left({}^{1}/_{k}\right) \sum_{q=1}^{k} \left(d_{i,q} - \bar{d}_i\right)^2}$ of inter-class IP-to-IP distance d_i describes the relative degree of closeness of IP In_{ij_c}, the ith IP of image j of class c with k inter-class matching distances.

The obtained values of standard deviation σ_{111} of In_{111} using five inter-class-IP distances *{d_i}* are *.1550.* Similarly, the standard deviation of In_{211}, In_{311} and In_{411} is σ_{211}*=.0541,* σ_{311}=.0466 and σ_{411}*=.0771,* respectively. Now, for each IP of training image *j,* In_{ij_c}, the corresponding threshold value θ_{ijc} can be calculated as $\theta_{ijc}=2\sigma_{ijc}$ and the threshold values of In_{111}, In_{211}, In_{311} *and* In_{411} are θ_{111}=.3100, θ_{211}=.1082, θ_{311}=.0932 and θ_{411}=.1542, respectively. This process is repeated for all training images. The meaning is for each IP of training image C_{11}, a threshold value θ_{ijc} is calculated. This process is repeated for all the images in the training set T so that every IP of each training image is associated with a threshold value, and the values are given in Table 2.

2.3 Determination of nearest interest point (NIP)

The second step in the proposed method is to calculate the *IP-to-IP* distances *di* between the test image Z and each of the training images using (2) as shown in Fig. 3a, b. Table 2 shows 5 *IPs* produced by the test image Z that are represented as *InZ_1, InZ_2, InZ_3, InZ_4* and *InZ_5.* As mentioned before, the number of SURF IPs generated from each image varies. The matching pair of IPs and the corresponding distances with training images are also given. There are only 4 IPs for C_{11} and so 4 *di* values are produced



matching pair
(IP of image C11, IP of image Z)

In_{111}	In_{Z-1}
In_{211}	In_{Z-3}
In_{311}	In_{Z_5}
In_{411}	In_{Z_4}

Fig. 3 Matching pair formation based on minimum distance. **a** 4 IPs of image C_{11} and 5 IPs of image Z are used for matching. First IP of C_{11} produced minimum distance with first IP of Z which is considered as best match. The matched IPs are represented in solid line to form matching pair (In_{111}, In_{Z-1}). Similarly, all the 4 IPs of C_{11} form matching pair with IPs of Z. **b** IPs matched between IPs of image C_{11} and IPs of image Z

($di(In_{111}, InZ\_1)=.1069, di(In_{211}, InZ\_3)=.236, di\ (In_{311}, InZ\_5)=.0083$ *and* $di(In_{411}, InZ\_4)=.317$) between the test image *Z* and C_{11}. Similarly, 5,4,4,5,5,3 and 5 *di* values are produced between the training images $C_{12}, C_{13}, C_{21}, C_{22}, C_{23}, C_{31}$ and C_{32} and test image *Z*, respectively. In this instance, as a matching pair consists of an IP from the training image and an IP from test image, the corresponding distance *di* gives the measure of closeness of *IP* of training image with the test image *Z*. That means, the set of distances $\{di\}$ of each training image with test image decide the closeness of training image with the test image. This process gives local decision votes for *NIP*s. The classifier which classifies the *IP*s based on *NIP* is termed as *NIP* classifier.

Table 3 Class similarity score used for global decision

Collective class similarity scores, $S_g(c)$				Class probability score for each rank S_p		
Class, c	Rank			Rank		
	1	2	3	1	2	3
C1	1	1.75	2.25	1/1 = 1	1.75/1.75 = 1	2.25/2.25 = 1
C2	0.25	0.45	0.45	0.25/1 = 0.25	0.45/1.75 = 0.257	0.45/2.25 = 0.2
C3	0.6	0.6	–	0.6/1 = 0.6	0.6/1.75 = 0.34	

Definition 2 An IP In_{ijc}, belonging to a set of M interest points ($i = 1,\ldots, M$) of a training image *j,* is classified as one of the NIPs of the given training image *j*, if the minimum distance *di*, between the IP of the test image and IP of the training image *j*, In_{ijc}, is smaller than the corresponding threshold value $\theta_{\mathrm{ijc}} = 2\sigma_{ijc}$, where σ_{ijc} is the standard deviation of the inter-class IP-to-IP distances of IP In_{ijc}.

Let us demonstrate the concept of NIP detection based on minimum distances, *di,* and the inter-class thresholds $\theta_{ijc}= 2\sigma_{ijc}$ from the training images. There are 5 IPs generated from the test image *Z*. Now, the minimum distance between In_{111} and IPs of *Z*, $d_1 = .1069$, is calculated based on (2) and the matching pair formed is (In_{111}, *InZ_1*). Similarly, d2 = .2360, d3 = .0083 and d4 = .3170 are calculated for IPs In_{211}, In_{311} and In_{411} to form matching pairs (In_{211}, *InZ_3*), (In_{311}, *InZ_5*) *and* (In_{411}, *InZ_4*). Out of the 5 *IPs* of Z, 4 *IPs* produced matching pair with 4 *IPs* of training image C_{11}. We can observe that one of the IPs from *Z* has not participated in making matching pair as there are only 4 IPs in C_{11}. The IPs of training image *C11*, having smaller distance *di* than threshold values θ_{ijc}, are considered as NIPs. As $(d_1 = .1069) < (\theta_{111} = .3100)$, In_{111} is considered as NIP and $d2 > \theta_{211}$, so In_{211} is not considered as NIP. Similarly, In_{311} is NIP since $d3 < \theta_{311}$ and In_{411} is not NIP since $d4 < \theta_{411}$.

The process of thresholding the distances to determine the *NIPs* can be called *local voting*, where the *vote* (*vi*) values are

$$vi(NIP) = \begin{cases} 1 & if\ di \le \theta ijc, \\ 0 & otherwise \end{cases} \tag{3}$$

The voting decision (3) says, In_{ijc} with threshold value θijc, of training image *j* is considered as NIP, with *vote vi* = *1*, if the distance produced, *di,* between *IPs* in the matching pair (In_{ijc}, *InZ_i*) is less than the threshold value θijc. The selection of NIPs through local decision is represented by these votes using numerical values, as shown in Table 2.

2.4 Global decision through *NIP classifier*

For each of the training image in *T*, the *image similarity score* is obtained by adding up the *votes* representing the *NIPs* as illustrated in the column labeled 'image similarity score' in Table 2, and are normalized in the next column. To make the global decision about the class of the test image *Z*, the *top-ranked* technique is used in the proposed method. In this technique, the image with the maximum similarity score in individual class (the top-ranked image in each class) is taken to denote the whole class. That means, the top similarity score of the image in each class in training set from Table 2 is considered as the value of the *collective class similarity score* $S_g(c)$ at Rank-1 in global level and is shown in Table 3.

Normalize the collective class similarity score $S_g(c)$ by dividing with the maximum similarity score across the classes to obtain *class probability scores* S_p (Table 3). The class c*of the test image *Z* can be identified by finding the class having the highest value of the S_p as follows:

$$c^* = \arg\max S_p(c) \tag{4}$$

Although the simplest approach to make a global decision is top-rank technique, there may be various situations where the top rank represents multiple classes. In order to improve the global decisions, the image similarity scores from the lower ranks are taken into account and a simple addition is performed. To do this, image similarity scores (Table 2) of the first and second rank values of each class can be added to obtain the Rank 2 collective class similarity score. This process is repeated for Rank 3. The class probability scores, S_p, for each rank can be achieved through normalizing the rank-wise collective class similarity scores by dividing with the maximum collective class similarity scores, across the classes. A tie in top-rank classification is solved by adding the S_p values from the next lower ranks, to represent the similarity score for the class. By using lower ranks in score calculations, the inconclusiveness of class selection obtained from tie can be resolved. We can observe that class probability score for Class C1 at Rank 1 gives highest value 1 with no tie. However, the maximum number of ranks that are used for classification is limited to 3 in our proposed method as SURF features are large in numbers and a good accuracy is obtained till Rank 3 in our NIP classifier. The various steps of NIP classifier are given in Algorithm 1.

```
Algorithm 1: NIPs classifier
Training stage:
Requires: Consider the training set with extracted SURF features, where IPs of the training image j having a class
label c be $\{G = \{In_{pj_c}, \forall p, j, c \in [1, X], X \in Integer\}$ and $In_{ij_c}$ is $i^{th}$ IP of image j (there is a total of
x classes and a maximum of y images per class). The number of IPs of each image varies from image to image.
Consider image j of class c and image n of class $\bar{c}$ in the training set where $\bar{c}$ is any class other than c.
  1. Calculate inter class IP-to-IP distances: Let $In_{ij_c}$ be the $i^{th}$ IP of image j of class c.
     a) Find distance di using the equation (2) and form a matching pair ($In_{ij_c}$, $In_{mn_{\bar{c}}}$) where $In_{mn_{\bar{c}}}$ the
        $m^{th}$ IP of the image n of class $\bar{c}$.
     b) Set Distance {di} =di.
     c) Repeat steps a and b for all the inter-class images in the training set with class
        $\bar{c}$ where $\bar{c}$ is any class other than c
     The set of distances {di} for $In_{ij_c}$, from step (c) gives inter-class IP-to-IP distances of the $In_{ij_c}$ in
     image j.
  2. Calculation of threshold value $\theta_{ijc}$
     a) Calculate standard deviation $\sigma_{ijc}$ from the set of {di} values using Definition 1.
     b) Determine the threshold value using $\theta_{ijc} = 2\sigma_{ijc}$ using Definition 2 which gives the associated
        threshold value $\theta_{ijc}$ for $In_{ij_c}$.
  3. Repeat step 1 and 2 for all the IPs of image j in the training set to generate associated threshold values
     for IPs of image j.
  4. Repeat steps 1 to 3 for all the images of training set to produce threshold values for their IPs.
Testing stage:
Requires: Consider the training set with extracted SURF features, where IPs of the training image j having a class
label c be $\{G = \{In_{pj_c}, \forall p, j, c \in [1, X], X \in Integer\}$ and $In_{ij_c}$ is $i^{th}$ IP of image j from class c (there
is a total of x classes and a maximum of y images per class); threshold values $\theta_{ijc}$ of each IP of the images from
the training set; a test image Z, whose class is unknown.
  1. For each of the training image j, calculate matching pairs (InZ_i,$In_{ij_c}$) of IPs from test image Z, with
     distances {di} using the equation (2) as in Fig 3.
  2. For each training image j, find NIPs by selecting IPs of j with threshold $\theta_{ijc}$, whose distance di fall within
     the threshold $\theta_{ijc}$. Each NIP of the training image j is assigned a vote $v_i$=1 using (3).
  3. Image similarity scores for each training image is calculated by counting the total number of votes
     $v_i$=1 (NIP) associated with that image, as in Table 2.
  4. From each class, add the image similarity score of 'r' top-ranked training images to obtain $r^{th}$ rank
     collective class similarity score, $S_g$(c).
  5. Normalize the collective class similarity score by dividing with the maximum score of each rank
     across the classes to obtain class probability scores $S_p$. The class of the training image with highest score
     $S_p$ at top-rank (r=1) is assigned as the class of the test image Z. if any tie on decision at r=1, $S_p$ , for r=2 is
     considered.
```

3 Parameter selection

The most important parameter used in our proposed method was threshold value θ_{ijc}, on which local decision on NIP depends. σ_{ijc} is the standard deviation of inter-class *IP-to-IP* distances of In_{ij_c} of image *j* from class *c* in the training set *T*. That means the value of σ_{ijc} is determined by the IPs of the SURF descriptor generated from the mages in the database. The problem of optimizing θ_{ijc} becomes optimizing the values of $2\sigma_{ijc}$. Figure 4a shows the impact of threshold σ_{ijc} variation on recognition performance of the proposed method. It is observed that low value of threshold σ_{ijc} reduces the recognition accuracy, while the recognition accuracy reached its peak at around $2\sigma_{ijc}$. Consider the inter-class distances obtained from all the IPs in a training set as the values of a random variable having probability density function $f_X(x)$ and the standard deviation σ_e. Similarly, consider the intra-class distances attained from all the IPs in a training set as the values of a random variable having probability density function $f_Y(y)$, and the standard deviation σ_a. Then, X/σ_e and Y/σ_a are normalized inter-class and normalized intra-class distances, respectively, with the corresponding probability density functions $f_{Y/\sigma_a}(Y/\sigma a)$ and $f_{X/\sigma_e}(X/\sigma e)$. We can see that inter-class distribution is shifted to the higher values of the normalized distance Fig. 4b.

The training phase is used to set the parameter, threshold value θ_{ijc} of each IP of training image. The classification accuracy in the Tamil handwritten character dataset from HP Labs, which corresponds to the threshold value $\theta_{ijc} = 2\sigma_{ijc}$, is 90.2%. Because the selection of θ_{ijc} in NIP classifier uses the class information, the implementation of NIP classifier can be viewed as supervised.

4 Recognition result and discussion

The proposed NIP classifier gave promising results in our experiments as it uses collective class similarity score and removes false similarities. In this section, the proposed NIP classifier is compared with several classifiers by providing standard dataset.

4.1 Recognition accuracy

The literature results from Table 1 say that classification accuracy is good for relatively small class problems, and it degrades for very large class problems. In most of the approaches, the data have been collected manually and limited to only few character classes [20, 22, 23, 26, 27]. Only few works have used HP Labs Tamil offline handwritten characters database, the only standard database available for Tamil HCR [28–32]. We can understand that Bhattacharya

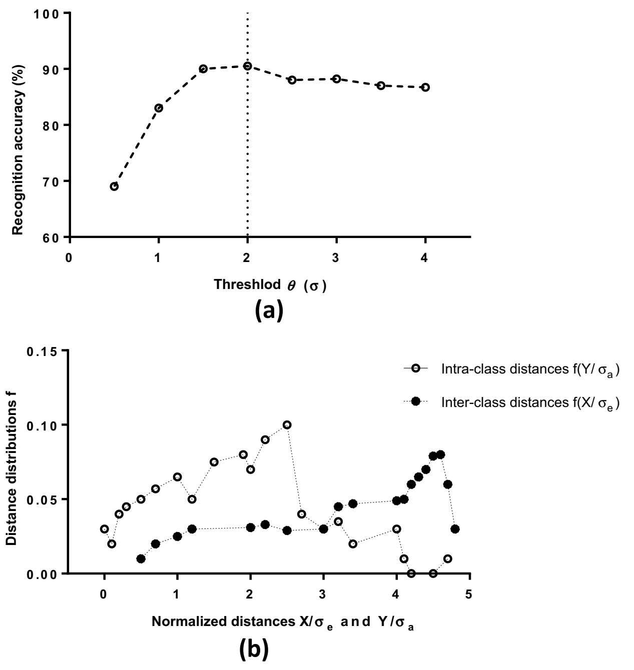


Fig. 4 Effect of parameters used in the NIP classifier. **a** Recognition accuracies for different values of θ(σ); **b** Inter-class and intra-class distributions of the normalized distances for the training set

et al. [29] and our previous paper 'Modified GA' [32] used all 156 character classes from HP Labs database, for training and testing. Among these two, the experiment in [32] involves only less number of samples from all the character classes and produced 89.5% accuracy, whereas [29] gave 89.6% accuracy on the complete samples of the database. But, the method, *'NIP classifier,'* was applied over the complete HP Labs Tamil offline handwritten characters database. Hence, the only comparison that we can perform is with the results of Bhattacharya et al. [29] who produced an accuracy of 89.6%. Our proposed method, *'NIP classifier,'* gives an accuracy of 90.2% with an improvement of 0.6% from [29], which is a significant enhancement in HCR area.

The proposed NIP classifier method gave promising results than the existing methods for Tamil HCR. The proposed method has been compared with classifiers such as k-NN, SVM, k-means, Brute-Force Matcher, FLANN, genetic algorithm, decision tree, Adaboost classifier, Bagging classifier, LogiBoost classifier and NNge classifier. It is found that our method outperforms in classification accuracy and is shown in Table 4.

4.2 Computational complexity

Bhattacharya et al. [29] used a two-stage classification approach as single-stage approach gave good results only for small class problems. The author used k-means clustering in stage 1 and MLP in stage 2 for classification.

Table 4 Recognition accuracies of the proposed method on Tamil dataset in comparison with other classifiers

Classifiers	NIP	Bhattacharya et al.	*k*-NN	*k*-means	SVM	Brute-Force Matcher	FLANN	Euclidean distance + genetic algorithm	Decision tree	Adaboost	Bagging	LogiBoost	NNge
Accuracy of HP Tamil Dataset	90.2%	89.6%	76.12%	71.03%	78.65%	77.95%	86.9%	87.3	82.7%	79.08%	83.15%	85.29	80%

The computational complexity of k-means clustering is O(n*k*I*d) where n is the number of samples, k is the number of clusters, I is the number of iterations and d is the number of features. The procedure was repeated till the classification error decreased to some quantity, leading to an increase in computation complexity. A set of groups of character classes are produced as the output of stage 1. Each group of character classes undergoes second-stage classification through MLP. 64-dimensional feature vector was used for input nodes, and the output nodes depend upon the number of classes in that group. The complexity of MLP depends upon number of input nodes, hidden nodes and number of output nodes. Author has considered 24 such groups in the second stage, and each group uses separate MLP for classification. We can understand that the complexity increases further in the second stage. It is also given that small group gave better accuracy and large group gave relatively less accuracy.

But, our proposed method has a computational complexity of $O(N(P\text{-}P1))$ where P is the total number of images and P1 is the number of images that does not have NIPs and N is the maximum number of IPs of image. Therefore, the computational complexity of *NIP classifier* is smaller in comparison with the combination of k-means clustering and MLP.

5 Conclusion

In this paper, an efficient offline Tamil HCR system has been proposed. The proposed *NIP classifier* introduces the concept of local feature decision in the first phase, and later global decision is taken to make a conclusion on the class of the test character image. Without compromising on the recognition accuracy, the proposed method simplifies the use of variable length as well as high dimensional feature vector which are measured as major issues of HCR systems. Benchmark database is used to demonstrate robust recognition performance. The state-of-the-art performance is achieved as NIP classifier calculates collective class similarity voting and thus reducing false recognitions. In contrast from conventional classification approaches, which depend on feature reduction methods, the result presented in this paper proves that the proposed classifier can successfully operate on the greater availability of features in the problems with high dimensional images. The future scope of our method is that the system can be further improved to reach the recognition accuracy of online character recognition systems.

Acknowledgements We extend our gratitude to HP Labs for providing database for our research.

References

1. Pal U, Chaudhuri BB (2004) Indian script character recognition: a survey. Pattern Recogn 37:1887–1899
2. Plamondon R, Srihari SN (2000) Online and off-line handwriting recognition: a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 22(1):63–84
3. Joshi N, Sita G, Ramakrishnan AG, Madhvanath S (2004) Comparison of elastic matching algorithms for online Tamil handwritten character recognition. In: Ninth international workshop on frontiers in handwriting recognition
4. Lorigo LM, Govindaraju V (2006) Offline Arabic handwriting recognition: a survey. IEEE Trans Pattern Anal Mach Intell 28(5):712–724
5. Subashini A, Kodikara ND (2011) A novel SIFT-based codebook generation for handwritten Tamil character recognition. In: 6th international conference on industrial and information systems
6. Kimura F, Takashina K, Tsuruoka S, Miyake Y (1987) Modified quadratic discriminant functions and the application to Chinese character recognition. IEEE Trans Pattern Anal Mach Intell. 1:149–153
7. Liu CL, Nakashima K, Sako H, Fujisawa H (2003) Handwritten digit recognition: benchmarking of state-of-the-art techniques. Pattern Recogn 36:2271–2285
8. Kim HY, Kim JH (2001) Hierarchical random graph representation of handwritten characters and its application to hangul recognition. Pattern Recogn 34:187–201
9. Gillies AM, Hepp D, Gader PD (1992) A system for recognizing handwritten words. Technical report submitted to the United States postal service, office of advanced technology, Nov. 1992
10. The Unicode Consortium (2000) The Unicode Standard 3.0. Addison Wesley publishers, Harlow
11. BBC (2004) India sets up classical languages. August 17, 2004. http://news.bbc.co.uk/2/hi/south_asia/3667032.stm
12. The Hindu (2005) Sanskrit to be declared classical language. October 28, 2005. Retrieved on 2007-08-16. http://www.hindu.com/2005/10/28/stories/2005102809281200.htm
13. Isolated Handwritten Tamil Character Dataset, hpltamil-iso-char http://www.hpl.hp.com/india/research/penhw/resources/tamil-iso-char.html
14. Raj R, Antony M, Abirami S (2016) Offline tamil handwritten character recognition using statistical based quad tree. Aust J Basic Appl Sci 10(2):103–109
15. Sundaram S, Urala KB, Ramakrishnan AG (2012) Language models for online handwritten tamil word recognition. In: Proceeding of the workshop on document analysis and recognition, December 16–16, 2012, Mumbai, India
16. Connell SD, Jain AK (2001) Template-based online character recognition. Pattern Recogn 34:1–14
17. Kunwar R, Ramakrishnan AG (2011) Tamil online handwriting recognition using fractal features. Tamil Internet 2011, At University of Pennsylvania, USA
18. Sundaresan CS, Keerthi SS (1999) A study of representations for pen based handwriting recognition of Tamil characters. In: Proceedings of the fifth international conference on document analysis and recognition
19. Aparna KH, Subramanian V, Kasirajan M, Prakash GV, Chakravarthy VS, Madhvanath S (2004) Online handwriting recognition for Tamil. In: Ninth international workshop on frontiers in handwriting recognition
20. Chinnuswamy P, Krishnamoorthy SG (1980) Recognition of hand printed tamil characters. Pattern Recogn 12:141–152
21. Suresh RM, Arumugam S, Ganesan L (1999) Fuzzy approach to recognize handwritten Tamil characters. In: Proceedings third

international conference on computational intelligence and multimedia applications

22. Hewavitharana S, Fernand HC (2002) A two stage classification approach to tamil handwriting recognition. Tamil Internet, California
23. Sutha J, Ramaraj N (2007) Network based offline Tamil handwritten character recognition. In: System international conference on computational intelligence and multimedia applications
24. Wahi A, Sundaramurthy S, Poovizhi P (2013) Handwritten Tamil character recognition. In: Fifth international conference on advanced computing
25. Kannan RJ, Prabhakar R (2008) Off-line cursive handwritten Tamil character recognition. WSEAS Trans Signal Process Arch 4(6):351–360
26. Pal U, Sharma N, Wakabayashi T, Kimura F (2008) Handwritten character recognition of popular south indian scripts. In: Doermann D, Jaeger S (eds) Arabic and chinese handwriting recognition. SACH 2006. Lecture Notes in Computer Science, vol 4768. Springer, Berlin, Heidelberg
27. Shanthi N, Duraiswami K (2010) A novel SVM-based handwritten Tamil character recognition system. Pattern Anal Appl 13(2):173–180
28. Vijayaraghavan P, Misha S (2015). Handwritten Tamil recognition using a convolutional neural network. NEML Poster 2015
29. Bhattacharya U, Ghosh SK, Parui SK (2007) A two stage recognition scheme for handwritten Tamil characters. In: Proceedings of the ninth international conference on document analysis and recognition (ICDAR 2007). IEEE Computer Society, Washington, DC, 511–515
30. Ashlin Deepa RN, Rao RR (2016) An efficient offline Tamil handwritten character recognition system using zernike moments and diagonal-based features. Int J Appl Eng Res 11(4):2607–2610
31. Ashlin Deepa RN, Rao RR (2017) An eigen characters method for recognition of handwritten tamil character recognition. In: Proceedings of the first international conference on intelligent computing and communication, advances in intelligent systems and computing
32. Ashlin Deepa RN, Rao RR (2017) A modified GA classifier for offline Tamil handwritten character recognition. Int J Appl Pattern Recognit 4(1):89–105
33. Bharath A, Madhvanath S (2012) HMM-based lexicon driven and lexicon-free word recognition for online handwritten indic scripts. IEEE Trans Pattern Anal Mach Intell 34(4):670–682
34. John J, Pramod KV, Balakrishnan K (2011) Handwritten character recognition of South Indian scripts: a review. In: National conference on Indian language computing, Kochi, Feb 19–20
35. Paulpandian T, Ganapathy V (1993) Translation and scale invariant recognition of handwritten Tamil characters using hierarchical neural networks. In: IEEE international symposium on circuits and systems
36. Jain Anil K, Taxt Torfinn (1996) Feature extraction methods for character recognition-a survey. Pattern Recognit 29(4):641–662
37. Bay H, Ess A, Tuytelaars T, Gool LV (2008) Speeded-up robust features (SURF). Comput Vis Image Underst 110(3):346–359
38. Lia A, Jianga W, Yuana W, Daia D, Zhanga S, Weia Z (2017) An improved FAST + SURF fast matching algorithm. Proced Comput Sci 107:306–312
39. Vinay A, Vasukib V, Bhatb S, Jayanth KS, Murthya KNB, Natarajan S (2016) Two dimensionality reduction techniques for SURF based face recognition. Proced Comput Sci 85:241–248
40. Mehrotra H, Pankaj KS, Majhi B (2013) Fast segmentation and adaptive SURF descriptor for iris recognition. Math Comput Model 58:132–146
41. Li C, Khan L, Prabhakaran B, (2007). Feature selection for classification of variable length multiattribute motions. In: Multimedia data mining and knowledge discovery, pp 116–137
42. Bandyopadhyay S, Murthy CA, Pal SK (1998) Pattern classification using genetic algorithms: determination of H. Pattern Recognit Lett 19:1171–1181
43. Bhatia N (2010) Survey of nearest neighbor techniques. Int J Comput Sci Inf Secur, pp 302–305
44. Duda RO, Hart PG, Stork DE (2001) Pattern classification. Wiley, New York
45. Cover T, Hart P (1967) Nearest neighbour pattern classification. IEEE Trans Inf Theory 13:21–27
46. Bailey T, Jain A (1978) A note on distance-weighted k-nearest neighbour rules. IEEE Trans Syst Man Cybern 8:311–313
47. Dudani S (1976) The distance-weighted k-nearest-neighbor rule. IEEE Trans Syst Man Cybern 6:325–327
48. Beyer K, Goldstein J, Ramakrishnan R, Shaft U (1999) When is nearest neighbour meaningful? In: Proceedings of the 7th international conference of database theory ICDT 99, Lecture Notes in Computer Science, Jerusalem, Israel, January 10–12, pp 217–235
49. Houle ME, Kriegel HP, Krger P, Schubert E, Zimek A (2010) Can shared-neighbor distances defeat the curse of dimensionality? In: Proceedings of the SSDBM, pp 482–500
50. Guyon I, Elisseeff A (2003) An introduction to variable and feature selection. J Mach Learn Res 3:1157–1182
51. Peng H, Long F, Ding C (2005) Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans Pattern Anal Mach Intell 27(2005):1226–1238
52. Weinberger KQ, Saul LK (2009) Distance metric learning for large margin nearest neighbour classification. J Mach Learn Res 10:207–244
53. Xing EP, Ng AY, Jordan MI, Russell S (2002) Distance metric learning, with application to clustering with side-information. In: Advances in neural information processing systems NIPS 2001, Vancouver, Canada, December 10–12, pp 521–528
54. Goldberger J, Roweis S, Hinton G, Salakhutdinov R (2004) Neighbourhood component analysis. In: Advances in neural information processing systems NIPS, pp 513–520
55. James AP, Dimitrijev S (2012) Nearest neighbor classifier based on nearest feature decisions. Comput J 55:1072–1087
56. Zuo W, Zhang D, Wang K (2008) On kernel difference weighted *k*-nearest neighbor classification. Pattern Anal Appl 11:247–257
57. Shakhnarovich G, Darrell T, Indyk P (2006) Nearest-neighbor methods in learning and vision: theory and practice. MIT Press, Cambridge
58. Akkus A, Guvenir AH (1996) K nearest neighbor classification on feature projections. In: Proceedings of the ICML, Bari, Italy, July 3–6, pp 12–19
59. Demirz G, Guvenir AH (1997) Classification by voting feature intervals. In: Proceedings of the ECML-97, Prague, Czech Republic, April 23–25, pp 85–92. Springer
60. Zhang H, Liu G, Chow TWS, Liu W (2011) Textual and visual content-based anti-phishing: a Bayesian approach. IEEE Trans Neural Netw 22(10):1532–1546