An Efficient Iris code storing and searching technique for Iris Recognition using Non-homogeneous K-d tree

Kavitha Amit Bakshi
Dept., of Computer Science
BNMIT
Bengaluru, India
kavitha.bakshi@gmail.com

B.G.Prasad
Dept., of Computer Science
BMSCE
Bengaluru, India
bgprasad.cse@bmsce.ac.in

Sneha.K
Dept., of Computer Science
BNMIT
Bengaluru, India
sneha.k30@gmail.com

Abstract— Iris Recognition is increasingly being used as the main method for biometric authentication since it is highly reliable and accurate. It considers the unique patterns of the iris to identify personnel by applying pattern recognition techniques. As the usage of iris recognition system increases, the number of iris code to be stored and retrieved for matching in large database of iris increases proportionally. Searching for the iris match in a huge database of iris templates poses challenges to research community in terms of retrieval accuracy and efficiency. The non-homogenous K-d tree structure used in the proposed model stores and matches iris code, which improves the search accuracy of the iris recognition system. The proposed model is tested on IITD and CASIA datasets.

Keywords—Iris Recognition, K-d tree, non-homogeneous K-d tree

I. INTRODUCTION

Iris recognition is an emerging Biometric identification technique which makes it an important field to do more research in this field.

Many advancements are seen in Iris localization, Normalization and Feature extraction methods which has helped in accurately identifying the iris and encoding the iris patterns effectively. However, storage and retrieval of iris code for a large database poses a challenge. The iris matching time considerably increases as the iris templates database increases. The overall system performance might have adverse effect because of this.

This work presents a technique to store and retrieve iris code using non-homogeneous K-d tree structure. K-d tree is a binary tree with K dimensions or key values. Non-homogeneous K-d tree lets data to be stored in the external nodes which can be stored in secondary memory, thereby freeing up the primary memory. This helps in overcoming memory space constraints faced in homogenous K-d tree [3], where in all data are stored in primary memory. As the number of Iris templates increases, the performance of the Iris recognition system decreases. In non-homogeneous K-d tree, increase in database size has little or no effect on performance of the system. Also, non-homogeneous K-d tree helps in reducing the search space to particular buckets or bins. The response time for searching using non-homogeneous K-d tree is inversely proportional to the size of the iris database, i.e., as the size of the iris database increases the search time decreases.

There are mainly two processes in an Iris recognition system: Iris Learning and Iris Recognition processes. Iris learning involves initial feature extraction of the iris image and construction of K-d tree which stores each Iris code and the corresponding personal ID as a node in the K-d tree. Iris recognition involves feature extraction of the iris to be matched and traversal of the tree to find the match.

Few well-known algorithms for Iris localization, normalization and feature extraction such as Hough Transforms, Daughman’s Rubbersheet model and 1-D log Gabor filter [4] are used for investigational purposes respectively. Both homogeneous k-d tree and non-homogeneous K-d tree are implemented and compared in terms of efficiency and accuracy. The non-homogeneous K-d tree iris matching has shown considerable improvements in terms of search accuracy and memory usage, thereby increasing the overall system performance.

II. LITERATURE REVIEW

Many researches have shown that Iris recognition is best suited for biometric identification of personnel [4][5][6] and have developed automated iris recognition systems. The system consists of Iris Localization and Normalization which are preprocessing steps. After preprocessing, Feature extraction is performed. Finally, Storage and Iris matching processes, completes the iris recognition system.


For papers in which all authors are employed by the US government, the copyright notice is: U.S. Government work not protected by U.S. copyright

For papers in which all authors are employed by a Crown government (U.K, Canada, and Australia), the copyright notice is: 978-1-4673-9563-2/15/$31.00 ©2015 Crown

For papers in which all authors are employed by the European Union, the copyright notice is: 978-1-4673-9563-2/15/$31.00 ©2015 European Union

For all other papers the copyright notice is: 978-1-4673-9563-2/15/$31.00 ©2015 IEEE
Daugman [4] developed a rubbersheet model for normalizing the segmented iris which helps to eliminate any imaging inconsistencies while capturing the iris image. Peng Yao et al., [9] proposed feature extraction process using modified log-Gabor filters. It is a strictly bandpass filter which does better extraction of iris features. W.W Boles et al., [12] presented feature extraction process using wavelet transforms as the gradient directions works as discriminating texture feature. J.Daughman [8] proposed many new approaches to iris recognition with suggestions on improving the iris localization methods. Paul.A, et al., [17] examined the image compression technique on Iris images. Kevin W. Bowyer et al., [13] have carried out extensive survey on history and current trends in iris recognition techniques. Many techniques available have been compared along with their test results.

In this paper, the most popular methods such as Hough transforms, Daughman’s rubbersheet model and 1-D log Gabor filters are adopted for iris localization, normalization and feature extraction processes respectively.

Multidimensional binary tree [1][2] is shown to be efficient in its requirement of storage. It has an averaging running time for n samples as O(logn) and a guaranteed search time of O(nlogn). Steve Zhou et al., [3] has developed a novel technique to storing and matching of Iris codes generated from feature extraction process in a homogeneous K-d tree structure. In this paper, a novel approach for storage and retrieval of iris code is employed using non-homogeneous K-d tree data structure which proved to be better in recognition accuracy and efficient in terms of memory usage compared to homogeneous K-d tree technique.

III. PROPOSED METHODOLOGY

Fig.1 shows the Iris recognition system. Iris localization is performed at first using Hough transforms which segments the iris and pupil. It identifies the pupil and iris centers and radii. Eyelids and eyelashes are considered as noise and removed from the segmented iris image. The segmented iris is then normalized to 20 x 240 image size using Daughman’s rubbersheet model. 1-D log Gabor filter is used which is then quantized to get 9600 bits of iris code. The non-homogeneous K-d tree is used to store these iris codes.

A. Iris Localization

Iris localization involves applying the Gaussian filters to smooth the image. Canny edge detection is used to generate edge map. Later Hough transforms is used to find the pupil and iris. Eyelids and eyelashes are removed using Hough line detector.

B. Iris Normalization

In Iris normalization, interpolation is carried out to convert from Cartesian to Polar co-ordinates as shown below:

\[ I(x(r,q), y(r,q)) \rightarrow I(r,q) \]

where \( q \) varies from 0 to \( 2\pi \) and \( q = \frac{2\pi}{240}; \)

Normalized image of 20x240 is produced.

C. Feature Extraction

In Feature extraction process, each row of the normalized image is convolved with 1D log-Gabor filters. The binary iris code or iris template is generated which is of 9600 bits.

D. Construction of Non-homogeneous K-d tree

Non-homogeneous K-d tree structure is a binary tree which can have more than one key that is K dimensions [1]. Different types of queries are possible using K-d tree like Exact match query, Nearest neighbor query, Range query and other queries [2].

Two keys, key1 and key2 are considered for construction of the K-d tree. Key1 is the Iris code and Key2 is the personal ID number of an individual. Here, personal ID is an auto-generated number for testing purposes. The keys key1 and key2 are used alternatively at each level of the K-d tree as splitting criteria or decisive factor for generating the left or right child node. The K-d tree organizes the iris codes in equal size bins or buckets which are attached to the leaf nodes. 16 iris codes are stored in a bin. The bins are stored as separate files in secondary memory which resolves memory issue posed by homogeneous K-d tree, where all the information is stored in main memory [3].

Non-homogeneous K-d tree is constructed by creating a root node at the center with the median value of key1. In the next level, key2 is used for sorting and creating left and right nodes with the respective median values. This process repeats till maximum number of iris code remaining under the current leaf node is equal to 16. After this the bins are created consisting of 16 iris codes and the leaf nodes are updated with the respective bin numbers. Total number of bins in a K-d tree is equal to 2 to the power of number of tree level.

Algorithm: Construction of Non-homogeneous K-d tree
Input: Iris image path, N (total # of Iris images)
Output: K-d tree, Bins
Method:
1. Sort the items based on key1 – iris code
2. Create root node with the median value
3. Sort the items based on key2 – personal ID
4. Create the left child node with the median value from the sorted list
5. All the nodes on the left are values lesser than the key on previous level
6. Create the right child node with the median value from the sorted list
7. All the nodes on the right are values greater than the key on previous level
8. Alternate key1 and key2 at each level of the tree
9. Repeat steps from 3-8 till items on leaf nodes equals 16
10. Create separate bins at each leaf node.

K-d tree is thus constructed with bins as output.

A sample K-d tree generated for 256 images is as shown in Fig.2 below:

Fig.2: Non-homogeneous K-d tree (Total images = 256)

E. Iris Matching using Non-homogeneous K-d tree

An iris test image is input to Iris recognition system for finding a match from the learned iris database. The K-d tree is traversed till the leaf node with the 2 keys, key1 and key2. Corresponding keys are verified at each level which decides if next node to be visited is either left child or right child. If the value of key is less than the value in the current node then left child is chosen else right child is chosen. This process is repeated till it finds the leaf node with which a bin is attached. This particular bin file is loaded and each of the iris codes in the bin is matched with the test iris image. Calculation of distance measure between 2 Iris codes is done using Hamming distance to check for the match. If the value is less than the threshold value then it is considered a valid match. The recognition process is declared success if the personal IDs of the test iris image and the learned iris in the database is same and the hamming distance is less than the threshold value. Otherwise, it is not a valid match.

Algorithm: Iris Matching using Non-homogeneous K-d tree
Input: Test iris image, K-d tree, Bin

Output: Match Result
Method:
1. Preprocess the test iris image and generate the iris code through feature extraction.
2. Compare the test iris image key with the key value in the K-d tree node
3. Select the next node to traverse as left child if the test key value is lesser than the key value of the node in the previous level.
4. Select the next node to traverse as right child if the test key value is greater than the key value of the node in the previous level.
5. If the key values are same, then calculate the hamming distance of the 2 iris codes.
6. If the Hamming distance is less than threshold, then Match successful else its Match fail.
7. Continue till leaf node is reached, alternating key1 and key2 for comparison at each level of the tree.
8. Load the particular bin and match each iris code with the test iris code by calculating the hamming distance between the iris codes.
9. If the Hamming distance is less than threshold, then Match successful else its Match fail.

Non-homogeneous K-d tree utilizes memory better than homogeneous K-d tree. All the iris codes need not be available in the main memory at all times. Non-homogeneous K-d tree classifies the iris codes into different bins and saves them in secondary memory. Only the required bin is loaded onto the main memory. The bin to be loaded into main memory is selected by traversing the K-d tree till the leaf node where the bin number is specified.

Non-homogeneous K-d tree reduces the search space to very few iris codes, rather than comparing with many or all of the iris codes in the database. For example, it requires maximum of 19 comparisons for iris database of 256 images and 20 comparisons for 512 images in iris database. As the iris database increases the efficiency of search increases using the K-d tree structure. A sample K-d tree traversal for 256 images, where the test iris match is in Bin 3 is depicted in Fig.3.

Fig.3: Non-homogeneous K-d tree traversal to Bin3 (Total images = 256)
IV. EXPERIMENT RESULTS AND ANALYSIS

CASIA and IITD datasets are used for testing. 1024 images are considered and Iris codes for these images are generated after feature extraction. K-d tree is constructed and the Iris codes are stored in the tree. 1024 iris images of the same subjects learned earlier are selected and input as test iris image for Iris matching process. A total of 1024 images are processed and the number of comparisons done to check the accuracy and efficiency of non-homogeneous K-d tree structure in comparison to homogeneous K-d tree structure.

The performance of Homogeneous K-d tree and Non-homogeneous K-d tree are measured in terms of parameters such as:

- Search Accuracy
- Search Accuracy for varying Iris code size
- Memory Efficiency

A. Search Accuracy:

Search accuracy indicates the accuracy of the K-d tree in searching and retrieval of valid match. The proposed model – Non-homogenous K-d tree has resulted in better accuracy than the existing Homogeneous K-d tree as shown in the Fig.4:

![Fig.4: Search Accuracy](image)

The results show that search accuracy of Non-homogeneous K-d tree is around 99.34% whereas search accuracy of Homogeneous K-d tree is 98.5%.

B. Search Accuracy for varying Iris code size:

Search accuracy is checked for varying Iris code sizes such as 4800 bits, 7200 and 9600 bits. In all cases, non-homogeneous K-d tree displayed better accuracy than the homogeneous K-d tree.

Even for Iris code of size 4800 bits accuracy of non-homogeneous K-d tree is higher than homogeneous K-d tree. 1024 iris images were considered for testing purpose. Fig.5 depicts the Search accuracy for varying iris code sizes.

![Fig.5: Search Accuracy -Varying code size](image)

C. Memory Efficiency:

Apart from accuracy benefits, non-homogeneous K-d tree does not fill up main memory which may result in lower performance of the system as it uses secondary memory to store iris codes. Number of Iris code in main memory for Non-homogeneous K-d tree is only 79 iris codes whereas it is 1024 iris code for homogeneous K-d tree for 1024 iris images Fig.6 shows the percentage of main memory consumption for Iris codes of 512 and 1024 images by homogeneous K-d tree in comparison with non-homogeneous K-d tree.

![Fig.6: Memory Usage](image)

Memory efficiency indicates efficient usage of Main memory. Number of Iris code stored in Main memory by Homogeneous K-d tree is much greater than the Non-homogeneous K-d tree. The proposed model – Non-homogenous K-d tree has proven to be more memory efficient than the existing Homogeneous K-d tree. It does not store all the Iris codes in main memory, it loads the corresponding bin into main memory only when required. This significantly eases the memory requirements even as the size of Iris database increases.
V. CONCLUSION AND FUTURE ENHANCEMENTS

Non-homogeneous K-d tree as a data structure for storage and retrieval of iris codes works more accurately around 99.3% whereas search accuracy is 98.4% for homogeneous K-d tree. Memory usage and response time of the system using non-homogeneous K-d tree decreases as the iris database increases. This shows that non-homogeneous K-d tree data structure is easily scalable. Only 7.7% iris code is stored in main memory for Non-homogeneous K-d tree whereas 100% iris code is stored in main memory for homogeneous K-d tree for 1024 iris images. In non-homogeneous K-d tree, only the specific bin containing 16 iris codes is loaded onto main memory whenever required. Overall performance is thus maintained constant as the main memory will not become a bottleneck using Non-homogeneous K-d tree structure.

As future enhancements, many features of K-d tree can be further explored. K-d tree are known for its efficiency in various searching techniques like Range query, nearest neighbor query and partial match query. These techniques needs to be checked for its usage in image processing domain.

References

[9] Peng Yao; Jun Li; Xueyi Ye; Zhenquan Zhuang; Bin Li, "Iris Recognition Algorithm Using Modified Log-Gabor Filters," Pattern Recognition, 2006. ICPR 2006. 18th International Conference on , vol.4, no., pp.461,464, 0-0 0.