INDEXING OF LARGE BIOMETRIC DATABASE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor in Technology
In
Computer Science and Engineering

Madhurima Dharchaudhuri
Roll No 10606028

Under the guidance of

Prof Banshidhar Majhi

Department of Computer Science and Engineering

National Institute of Technology, Rourkela
May, 2010

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled, "**AN INDEXING TECHNIQUE FOR BIOMETRIC DATABASE**" submitted by Madhurima Dharchaudhuri in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Computer Science and Engineering at the National Institute of Technology, Rourkela, is an authentic work carried out by her under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university / institute for the award of any Degree or Diploma.

Date:

Prof Banshidhar Majhi

Dept. of Computer Science and Engineering

National Institute of Technology, Rourkela

Rourkela – 769008

Acknowledgement

In the first place, I am heartily thankful to my supervisor, Prof B. Majhi for his supervision, advice and guidance from the very early stage of this project till the last level as well as providing me encouragement and support in various ways which enabled me to develop an understanding of the subject. I owe my deepest gratitude to Miss Hunny Mehrotra that in the midst of all her activities she accepted to offer her best possible help.

I am indebted to many of my friends to have supported me and would like to convey my special thanks to a close friend Subhankar Mishra without whose cooperation and encouragement in the last stage I could never have produced this work. I sincerely thank my parents for being there with me throughout and inspiring me with strength and positive attitude whenever I was not very confident.

Lastly, I offer my regards to all professors and research scholars in the department who supported me in any respect during the completion of the project. I express my pleasure in thanking everybody who made this thesis possible and also my apology that I could not mention personally one by one.

Madhurima Dharchaudhuri

Roll No. 10606028
Computer Science and Engineering
NIT Rourkela

Abstract

The word "biometrics" is derived from the Greek words 'bios' and 'metric' which means life and measurement respectively. This directly translates into "life measurement**".** Biometrics is the automated recognition of individuals based on their behavioral and biological characteristics. Biometric features are information extracted from biometric samples which can be used for comparison with a biometric reference. Biometrics comprises methods for uniquely recognizing humans based upon one or more intrinsic physical or behavioral traits. In computer science, in particular, biometrics is used as a form of identity access management and access control. It is also used to identify individuals in groups that are under surveillance. Biometrics has fast emerged as a promising technology for authentication and has already found place in most hi-tech security areas. An efficient clustering technique has been proposed for partitioning large biometric database during identification. The system has been tested using bin-miss rate as a performance parameter. As we are still getting a higher bin-miss rate, so this work is based on devising an indexing strategy for identification of large biometric database and with greater accuracy. This technique is based on the modified B+ tree which reduces the disk accesses. It decreases the data retrieval time and also possible error rates. The indexing technique is used to declare a person's identity with lesser number of comparisons rather than searching the entire database. The response time deteriorates, as well as the accuracy of the system degrades as the size of the database increases. Hence for larger applications, the need to reduce the database to a smaller fraction arises to achieve both higher speeds and improved accuracy. The main purpose of indexing is to retrieve a small portion of the database for searching the query. Since applying some traditional clustering schemes does not yield satisfactory results, we go for an indexing strategy based on tree data structures. Index is used to look-up, input and delete data in an ordered manner. Speed and efficiency are the main goals in the different types of indexing. Speed and efficiency include factors like access time, insertion time, deletion time, and space overhead. The main aim is to perform indexing of a database using different trees beginning with Binary Search tree followed by B tree before proceeding to its variations, B+ tree and Modified B+ tree, and subsequently determine their performance based on their respective execution times.

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

TABLE OF CONTENTS

Chapter 1

1. INTRODUCTION

1.1 AN INTRODUCTION TO BIOMETRICS

WHAT IS BIOMETRICS?

"Biometrics" means "life measurement" but the term is usually associated with the use of unique physiological characteristics to identify an individual. Security is the application which most people associate with biometrics. However, biometric identification eventually has a much broader relevance as computer interface becomes more natural. Knowing the person with whom you are conversing is an important part of human interaction. The method of identification based on biometric characteristics is nowadays preferred over traditional passwords and PIN based methods for various reasons like the person to be identified is required to be physically present at the time-of-identification.

Biometrics utilize "something you are" to authenticate identification. This might include fingerprints, retina pattern, iris, hand geometry, vein patterns, voice password or signature dynamics. Biometrics can be used with a smart card to authenticate the user. The user's biometric information is stored on a smart card, the card is placed in a reader and a biometric scanner reads the information to match it against that on the card. This is a fast, accurate and highly secure form of user authentication.

Whether a human characteristic can be used for biometrics can be understood in terms of the following parameters:

- **Universality** – indicates that each person should have the characteristic.
- **Uniqueness** – means how well the biometric separates one individual from another.
- **Permanence** – measures how well a biometric resists aging and other variance over time.
- **Collectability** – refers to ease of acquisition for measurement.
- **Performance** – deals with accuracy, speed, and robustness of technology used.
- **Acceptability** – is degree of approval of a technology.
- **Circumvention** – is the ease of use of a substitute.
- **Measurability** – the properties should be suitable for capture without waiting time and must be easy to gather the attribute data passively.
- **Reducibility** – automated capturing and automated comparison with previously stored data requires that the captured data should be capable of being reduced to a file which is easy to handle.

- **Reliability and tamper-resistance** – the attribute should be impractical to mask or manipulate.
- **Privacy** – the process should not violate the privacy of the person.
- **Comparable** – should be able to reduce the attribute to a state that makes it digitally comparable to others. The less probabilistic the matching involved, the more authoritative would be the identification.
- **Inimitable** – the attribute must be irreproducible by other means. The less reproducible the attribute, the more authoritative the identification is.

A practical biometric system should meet the specified recognition accuracy, speed, and resource requirements, be harmless to the users, be accepted by the intended population, and be sufficiently robust to various fraudulent methods and attacks to the system [1].

A biometric system has two modes of operation viz.

- **Verification** – A one to one comparison of a captured biometric with a stored template to verify that the individual is the one who she claims to be. Verification can be done in conjunction with a smart card, username or ID number (based on "what she possesses" or "what she remembers").
- **Identification** – A one to many comparison of the captured biometric against a biometric database in an attempt to identify an unknown individual. The identification only succeeds in identifying the individual if the comparison of the biometric sample to a template in the database falls within a previously set threshold (based on "who she is").

The method consists of mainly three stages:

1. Data Acquisition
2. Feature Extraction
3. Matching

But this entire process is not that easy. There are few challenges that are faced by any biometric identification system [2]:

1. During identification the system has to operate on large dataset and the time taken by it to declare an identity should not be much.
2. To serve its purpose, an identification system needs an efficient searching and matching algorithm.
3. The number of false- positive in the system should not be very large as the size of the database increases.

A biometric system is a pattern recognition system that works in the following way: acquires biometric data from an individual, extracts a feature set from the acquired data, and compares this feature set against the template set in the database.

Biometrics commonly implemented or studied includes fingerprint, face, iris, voice, signature, and hand geometry. Many other modalities are in various stages of development and assessment. There is not one biometric modality that is best for all implementations. Biometrics are typically collected using a device called a *sensor*. These sensors serve two purposes: first they are used to acquire the data needed for recognition and then used to convert the data to a digital form. Example of "sensors" could be digital cameras (for face recognition) or a telephone (for voice recognition). A biometric *template* is a digital representation of an individual's distinct characteristics, representing information extracted from a biometric sample [3]. Biometric templates are what are actually compared in a biometric recognition system. The first time an individual uses a biometric system is called an *enrollment*. During the enrollment, biometric information from an individual is stored. In subsequent uses, biometric information is detected and compared with the information stored at the time of enrollment. If the biometric system is to be robust it is crucial that storage and retrieval of such systems themselves be secure. As we can see from the figure below, the first block (sensor) is the interface between the real world and the system; it has to acquire all the necessary data.

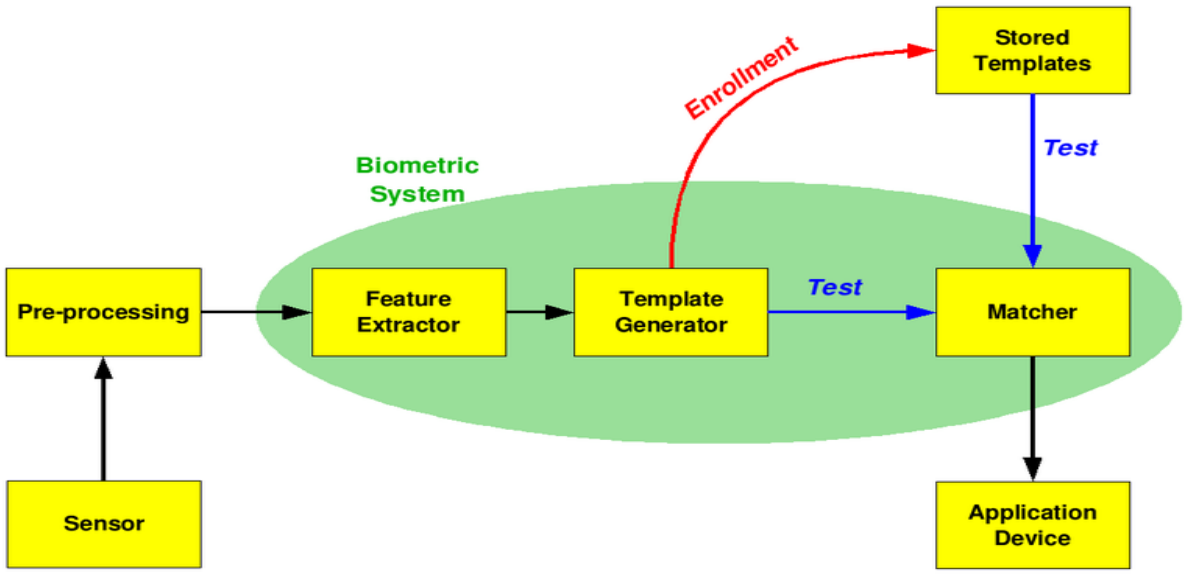


FIGURE 1: A GENERIC BIOMETRIC SYSTEM [4]

Most of the times it is an image acquisition system, but it can change according to the characteristics desired. Acquiring relevant data for the biometrics is one of the critical processes which have not received adequate attention. The amount of care taken in acquiring the data determines the performance of the entire system. Two of the tasks associated with it are [3]:

a. Quality assessment: Automatically assessing the suitability of the input data for automatic processing and

b. Segmentation: Separation of the input data into foreground (object of interest) and background (irrelevant information).

A number of opportunities exist for incorporating the context of the data capture which may further help improve the performance of the system and avoiding undesirable measurements (and subsequent recapture of desirable measurements). To detect the machine-readable representations completely capture the invariant and discriminatory information in the input measurements is the most challenging problem in representing biometric data. This representation issue constitutes the essence of system design and has far reaching implications on the design of the rest of the system. The unprocessed measurement values are typically not invariant over the time of capture and there is a need to determine salient features of the input measurement which both discriminate between the identities as well as remain invariant for a given individual. Thus, the problem of representation is to determine a measurement (feature) space which is invariant (less variant) for the input signals belonging to the same identity and which differ maximally for those belonging to different identities (high *interclass* variation and low *interclass* variation).

Referring Figure 1, the second block performs all the necessary pre-processing: it has to remove artifacts from the sensor, to enhance the input (e.g. removing background noise), to use some kind of normalization, etc. In the third block necessary features are extracted. This step is an important step as the correct features need to be extracted in the optimal way. A vector of numbers or an image with particular properties is used to create a template. A template is a synthesis of the relevant characteristics extracted from the source. Elements of the biometric measurement that are not used in the comparison algorithm are discarded in the template to reduce the file size and to protect the identity of the enrollee.

Given raw input measurements, automatically extracting the given representation is an extremely difficult problem, especially where input measurements are noisy. A given arbitrarily complex representation scheme should be amenable to automation without any human intervention. For instance, the manual system of fingerprint identification uses as much as a dozen features. However, it is not feasible to incorporate these features into a fully automatic fingerprint system because it not easy to reliably detect these features using state-of-the-art image processing techniques. Determining features that are amenable to automation has not received much attention in computer vision and pattern recognition research and is especially important in biometrics which are entrenched in the design philosophies of an associated mature manual system of identification. Traditionally, the feature extraction system follows a staged sequential architecture which precludes effective integration of extracted information available from the measurements. Increased availability of inexpensive computing and sensing resources makes it possible to use better architectures/methods for information processing to detect the features reliably. Once the features are determined, it is also a common practice to design feature

extraction process in a somewhat ad hoc manner. The efficacy of such methods is limited especially when input measurements are noisy. Rigorous models of feature representations are helpful in a reliable extraction of the features from the input measurements, especially, in noisy situations.

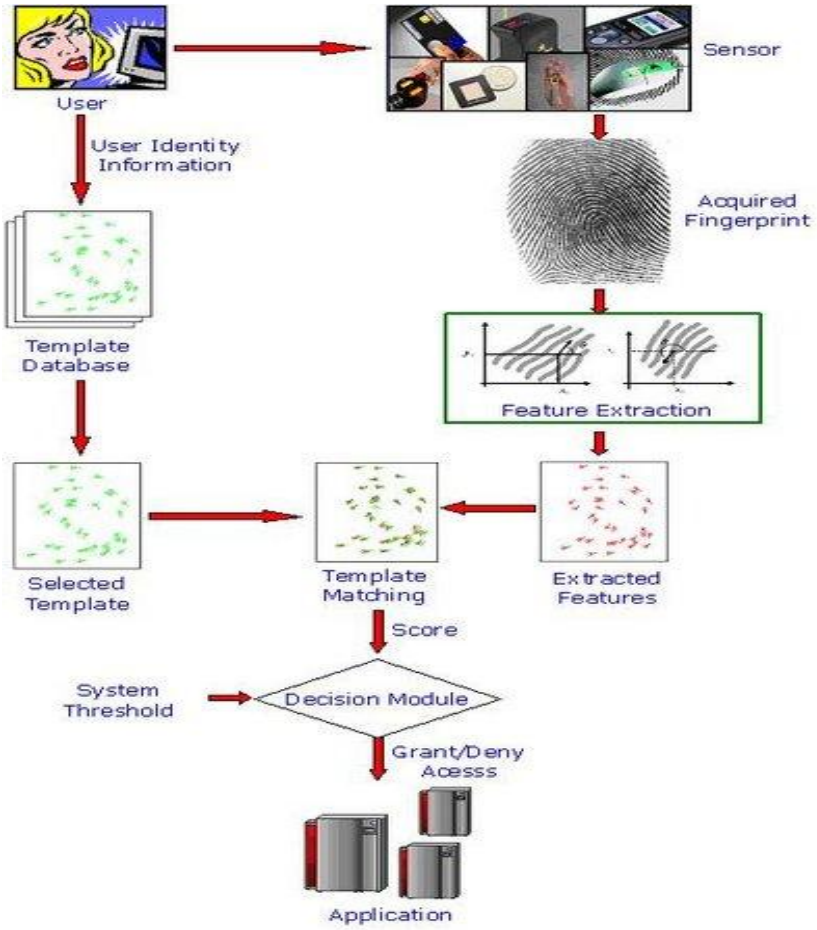


FIGURE 2: BIOMETRIC SYSTEM OPERATION [4]

If enrollment is being performed the template is simply stored somewhere (on a card or within a database or both). If a matching phase is being performed, the obtained template is passed to a matcher that compares it with other existing templates, estimating the distance between them using any algorithm. The matching program will analyze the template with the input. This will then be output for any specified use or purpose (e.g. entrance in a restricted area). The crux of a matcher is a similarity function which quantifies the intuition of similarity between two representations of the biometric measurements. Determining an appropriate similarity metric is a very difficult problem since it should be able to discriminate between the representations of two different identities despite noise, structural and statistical variations in the input signals, aging, and artifacts of the feature extraction module. In many biometrics, say signature verification, it is difficult to even define the ground truth: do the given two signatures belong to the same person

or different persons? A representation scheme and a similarity metric determine the accuracy performance of the system for a given population of identities; hence the selection of appropriate similarity scheme and representation is critical. Given a complex operating environment, it is critical to identify a set of valid assumptions upon which the matcher design could be based. Often, there is a choice between whether it is more effective to exert more constraints by incorporating better engineering design or to build a more sophisticated similarity function for the given representation. For instance, in a fingerprint matcher, one could constrain the elastic distortion altogether and design the matcher based on a rigid transformation assumption or allow arbitrary distortions and accommodate the variations in the input signals using a clever matcher. Where to strike the compromise between the complexity of the matcher and controlling the environment is an open problem.

Associating an identity with an individual is called personal identification. Human race has come a long way since its inception in small tribal primitive societies where every person in the community knew every other person. In today's complex, geographically mobile, increasingly electronically interconnected information society, accurate identification is becoming very important and the problem of identifying a person is becoming ever increasingly difficult. A number of situations require an identification of a person in our society: have I seen this applicant before? Is this person an employee of this company? Is this individual a citizen of this country? Many situations will even warrant identification of a person at the far end of a communication channel [5].

1.2 AN OVERVIEW OF BIOMETRICS TECHNOLOGY

1.2.1 TYPES OF BIOMETRICS

There are basically two types of biometrics:

1. Behavioral biometrics
2. Physical biometrics

Behavioral biometrics basically measures the characteristics which are acquired naturally over a time. It is generally used for verification.

Examples of behavioral biometrics include:

- Speaker recognition: which means analyzing vocal behavior
- Signature: deals with analyzing signature dynamics
- Keystroke: deals with measuring the time spacing of typed words

Physical biometrics measures the inherent physical characteristics of an individual. It can be used for either identification or verification.

Examples of physical biometrics include:

- Fingerprint: indicates analyzing fingertip patterns
- Facial recognition: refers to measuring facial characteristics
- Hand geometry: refers to measuring the shape of the hand
- Iris scan: mainly deals with analyzing features of colored ring of the eye
- Retinal scan: indicates analyzing blood vessels in the eye
- DNA: which means analyzing genetic makeup

It is not practically possible to have any single biometrics which is expected to satisfy the needs of all identification systems. Many of them have already been proposed, researched and evaluated. Each biometrics has its own strengths as well as limitations; and accordingly, each biometric appeals to a particular identification (authentication) application. The following describes few of the existing and burgeoning biometric technologies [4], [3]:

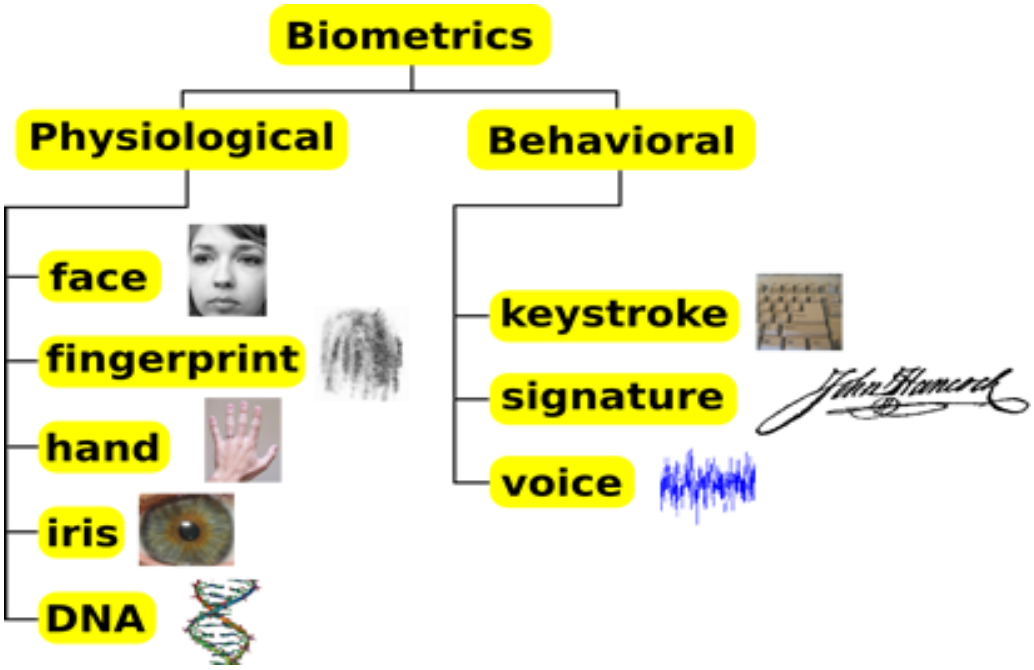


FIGURE 3: A CLASSIFICATION OF BIOMETRIC TRAITS [4]

VOICE

Our voices are unique to each person (including twins), and cannot be exactly replicated. Speech includes two components: a physiological component (the voice tract) and a behavioral component (the accent). It is almost impossible to imitate anyone's voice perfectly. Voice recognition systems can discriminate between two very similar voices, including twins. Voice is a characteristic of an individual. However, it is not expected to be sufficiently unique to permit identification of an individual from a large database of identities. A voice signal available for authentication is typically degraded in quality by the microphone, communication channel, and digitizer characteristics. Before extracting features, the amplitude of the input signal may be normalized and decomposed into several band-pass frequency channels. The features extracted from each band may be either time-domain or frequency domain features. Voice capture is unobtrusive and voice print is an acceptable biometric in almost all societies. Some applications entail authentication of identity over telephone. In such situations, voice may be the only feasible biometric. Voice is a behavioral biometrics and is affected by a person's health (e.g., cold), stress, emotions, etc. To extract features which remain invariant in such cases is very difficult. Besides, some people seem to be extraordinarily skilled in mimicking others. A reproduction of an earlier recorded voice can be used to circumvent a voice authentication system in the remote unattended applications. One of the methods of combating this problem is to prompt the subject (whose identity is to be authenticated) to utter a different phrase each time. Voice biometrics is mostly used for telephony-based applications. Voice verification is used for government, healthcare, call centers, electronic commerce, financial services, customer authentication for service calls, and for house arrest and probation-related authentication.

FINGERPRINTS

Fingerprint ridges are formed in the womb; we have fingerprints by the fourth month of fetal development. Once formed, fingerprint ridges are like a picture on the surface of a balloon. As the person ages, the fingers do get larger. However, the relationship between the ridges stays the same, just like the picture on a balloon is still recognizable as the balloon is inflated. Fingerprints are graphical flow-like ridges present on human fingers. Their formations depend on the initial conditions of the embryonic development and they are believed to be unique to each person (and each finger). Fingerprints are one of the most mature biometric technologies used in forensic divisions worldwide for criminal investigations and therefore, have a stigma of criminality associated with them. Typically, a fingerprint image is captured in one of two ways:

- Scanning an inked impression of a finger or
- Using a live-scan fingerprint scanner

Everyone is known to have their own unique and immutable fingerprints. Fingerprint matching techniques can be classified in two categories: minutiae based and correlation based. Both techniques have their own unique downfalls. Minutiae points are difficult to extract if the fingerprint scan quality is low. The correlation based method overcomes the downfalls encountered with the former method but requires the precise location of the registration points and are affected by image rotation and translation.

FACE

Face is one of the most acceptable biometrics because it is one of the most common method of identification which humans use in their visual interactions. In addition, the method of acquiring face images is non-intrusive. The dimensions, proportions and physical attributes of a person's face are unique. Biometric facial recognition systems will measure and analyze the overall structure, shape and proportions of the face: Distance between the eyes, nose, mouth, and jaw edges; upper outlines of the eye sockets, the sides of the mouth, the location of the nose and eyes, the area surrounding the cheekbones. The main facial recognition methods are: feature analysis, neural network, eigenfaces, and automatic face processing. Applications of face biometrics are access to restricted areas and buildings, banks, embassies, military sites, airports and law enforcements.

INFRARED FACIAL AND HAND VEIN THERMOGRAMS

The image is obtained by sensing the infrared radiations from the face of a person. The gray level at each pixel is characteristic of the magnitude of the radiation. Human body radiates heat and the pattern of heat radiation is a characteristic of each individual body. An infrared sensor could acquire an image indicating the heat emanating from different parts of the body. These images are called thermograms. The method of acquisition of the thermal image unobtrusively is akin to the capture of a regular (visible spectrum) photograph of the person. Any part of the body could be used for identification. The technology could be used for covert identification solutions and could distinguish between identical twins. It is also claimed to provide enabling technology for identifying people under the influence of drugs: the radiation patterns contain signature of each narcotic drug. A related technology using near infrared imaging is used to scan the back of a clenched fist to determine hand vein structure .Infrared sensors are prohibitively expensive which is a factor inhibiting wide spread use of thermograms.

IRIS

The iris is the elastic, pigmented, connective tissue that controls the pupil. The iris is formed in early life in a process called morphogenesis. Once fully formed, the texture is stable throughout life. It is the only internal human organ visible from the outside and is protected by the cornea.

The iris of the eye has a unique pattern, from eye to eye and person to person. An iris image is typically captured using a non-contact imaging process .The image is obtained using an ordinary CCD camera with a resolution of 512 dpi. Capturing an iris image involves cooperation from the user, both to register the image of iris in the central imaging area and to ensure that the iris is at a predetermined distance from the focal plane of the camera. A position-invariant constant length byte vector feature is derived from an annular part of the iris image based on its texture. An iris scan will analyze over 200 points of the iris, such as rings, furrows, freckles, the corona and will compare it with a previously recorded template. The identification error rate using iris technology is believed to be extremely small and the constant length position invariant code permits an extremely fast method of iris recognition. Applications of iris biometrics include: Identity cards and passports, border control and other Government programmes, prison security, database access and computer login, hospital security, schools, aviation security, controlling access to restricted areas, buildings and homes.

EAR

It is known that the shape of the ear and the structure of the cartilegenous tissue of the pinna are distinctive. The features of an ear are not expected to be unique to each individual. The ear recognition approaches are based on matching vectors of distances of salient points on the pinna from a landmark location on the ear. A new type of ear-shape analysis could see ear biometrics surpass face recognition as a way of automatically identifying people, claim the UK researchers developing the system. According to a biometrics expert at the University of Southampton, ears are remarkably consistent; unlike faces they do not change shape with different expressions or age, and remain fixed in the middle of the side of the head against a predictable background! Ears have been used to identify people before now, but other methods have used an approach similar to face recognition. This involves extracting key features, such as the position of the nose and eyes - or in the case of the ear, where the channels lie. These are then represented as a vector, describing where features appear in relation to each other. The new approach instead captures the shape of the ear as a whole and represents this in code, allowing the whole ear shape to be compared.

GAIT

Gait is the peculiar way one walks and is a complex spatio-temporal behavioral biometrics. Gait is not supposed to be unique to each individual, but is sufficiently characteristic to allow identity authentication. Gait is a behavioral biometric and may not stay invariant especially over a large period of time, due to large fluctuations of body weight, major shift in the body weight (e.g., waddling gait during pregnancy , major injuries involving joints or brain (e.g., cerebellar lesions in Parkinson disease), or due to inebriety (e.g., drunken gait). Gait biometrics identifies a person by the way they walk, run or any other type of motion of the legs; gait biometrics can be

used to identity everything from the length and thickness of an individual's legs to the stride of their step. Unlike other more researched and identifiable methods of biometrics, gait biometric technology faces the difficulty of identifying not only a particular body part but a motion. Its most important application: gait biometrics would be particularly beneficial in identifying criminal suspects.

KEYSTROKE DYNAMICS

Each person types on a keyboard in a characteristic way. This behavioral biometric offers sufficient information to allow identity verification though it is not expected to be unique to each individual. The keystrokes of a person using a system could be monitored unobtrusively as that person is keying in other information. Keystroke dynamic features are based on time durations between the keystrokes. Some variants of identity authentication use features based on inter-key delays as well as dwell-times (how long a person holds down a key). Typical matching approaches use neural network architecture to associate identity with the keystroke dynamics features.

SIGNATURE

Signatures are also a behavioral biometric that change over a period of time and are influenced by physical as well as emotional conditions of the signatories. The manner in which a person signs his or her name is a characteristic feature of that particular individual. Signatures of some people vary substantially. Apart from that, professional forgers may be able to reproduce signatures to fool the system.

1.2.2 PERFORMANCE MEASURES

For evaluating the efficiency of a biometric system, the following parameters are used:

- False Acceptance Rate (FAR)

  The frequency with which a non-authorized person is accepted as authorized is termed as FAR. It is generally a security relevant measure because a false acceptance can often lead to severe damages. It is a non-stationary statistical quantity which not only shows a personal correlation, but can even be determined for each individual biometric characteristic. This is known as personal FAR.

- False Rejection Rate (FRR)

  It is defined as the frequency with which an authorized person is rejected access. A false rejection mostly causes annoyance, so this is generally regarded as a comfort criterion. Similar to FAR, FRR is also a non-stationary statistical quantity. It does not only show a

strong personal correlation, but can also be determined for each individual biometric characteristic. This is known as personal FRR.

- Failure To Enroll rate (FTE or FER)

It indicates the proportion of people who fail to be enrolled successfully. This also is a non-stationary statistical quantity which not only shows a strong personal correlation, but along with that, it can be determined for each individual biometric characteristic. This property is called personal FER.

- Failure To Acquire (FTA) rate

Users who are enrolled but yet are mistakenly rejected after many identification/verification attempts count for this rate. It can originate through features which are temporarily not measurable. It is usually considered within the FRR and need not be calculated separately.

- False Identification Rate (FIR)

During an identification, the probability that the biometric features are falsely assigned to a reference is referred to as the False Identification Rate. The exact definition depends on the strategy of assignment; for example, after feature comparison, often more than a single reference exceeds the decision threshold.

- False Genuine Error or False Match (FM) is when the algorithm or identification method classifies as genuine an actual impostor comparison. The system incorrectly declares a successful match between the input pattern and a non-matching biometric template stored in the database, in the case of identification, or a template associated with an incorrectly claimed identity, in the case of verification. False Impostor Error or False Non Match (FNM) is when the algorithm classifies as impostor an actual genuine comparison. Here the biometric system incorrectly declares a failure of match between the input pattern and a matching pattern stored in the biometric database (in case of identification) or the pattern associated with the correctly claimed identity (in case of verification).

1.2.3 CHALLENGES FACED BY A BIOMETRIC SYSTEM

Biometrics is yet not a foolproof method of automatic human recognition, in spite of the fact that it appears to be the obvious technology for robust personal identification. Inexpensive and compact biometric sensors and fast processing chips are available now and with these it is becoming increasingly clear that a broader use of biometric technology would require better

solutions. But there are three fundamental barriers which it has to overcome. They are as mentioned below [1]:

- Recognition "performance" – it deals with "how to effectively represent and recognize biometric patterns?" For example, how to recognize a person with an accuracy of 99.999%? Research is still going on to improve the performance of various biometric recognition systems with the help of better feature representation techniques and matching algorithms. "Multibiometrics" is a technique to improve performance. It combines multiple biometric traits such as fingerprint, iris, etc. and such a system aims to effectively fuse the salient information among the individual biometric traits. This helps translate into better recognition performance in biometric systems.
- System "security" – it refers to "how to guarantee that the biometric systems are not vulnerable to disruption?" For example, is it possible to ensure that fraudsters cannot infiltrate the system? The security of biometric systems is indeed crucial. Assuring that the input biometric sample was actually presented by its legitimate owner, and the input pattern was actually matched by the system with the genuinely enrolled pattern samples require special attention. There are a number of ways a perpetrator may attack a biometric system; and two very solemn criticisms against biometric systems that have till date not been addressed agreeably are: 1. Biometrics are not secrets which implies that an attacker has ready access to a legitimate biometric trait and therefore could fraudulently enter it to gain access into the biometric system, and 2. Enrolled biometric templates are not revocable which implies that when a biometric trait has been compromised, the legitimate can in no way revoke the trait. Both these issues are being addressed by researchers.
- "Privacy" issues – it deals with "how to make sure that the biometric system is being exclusively used for the specified purpose?" The two main concerns of the users of biometric systems are: will the undeniable proof of biometrics-based access be used to track the person in such a way so as to contravene upon his right to privacy?, will the biometric data be abused for a purpose that is not intended? But as far as today, there are no acceptable solutions that can address the entire spectrum of privacy issues.

1.2.4 WHY BIOMETRICS?

There are several reasons why biometrics has become so popular:

- It is the most definitive real-time tool available today
- It can be combined with other tools to form more secure, easier to use verification solutions
- It recognizes individuals definitively
- It offers enhanced security and convenience over traditionally used identity governance tools

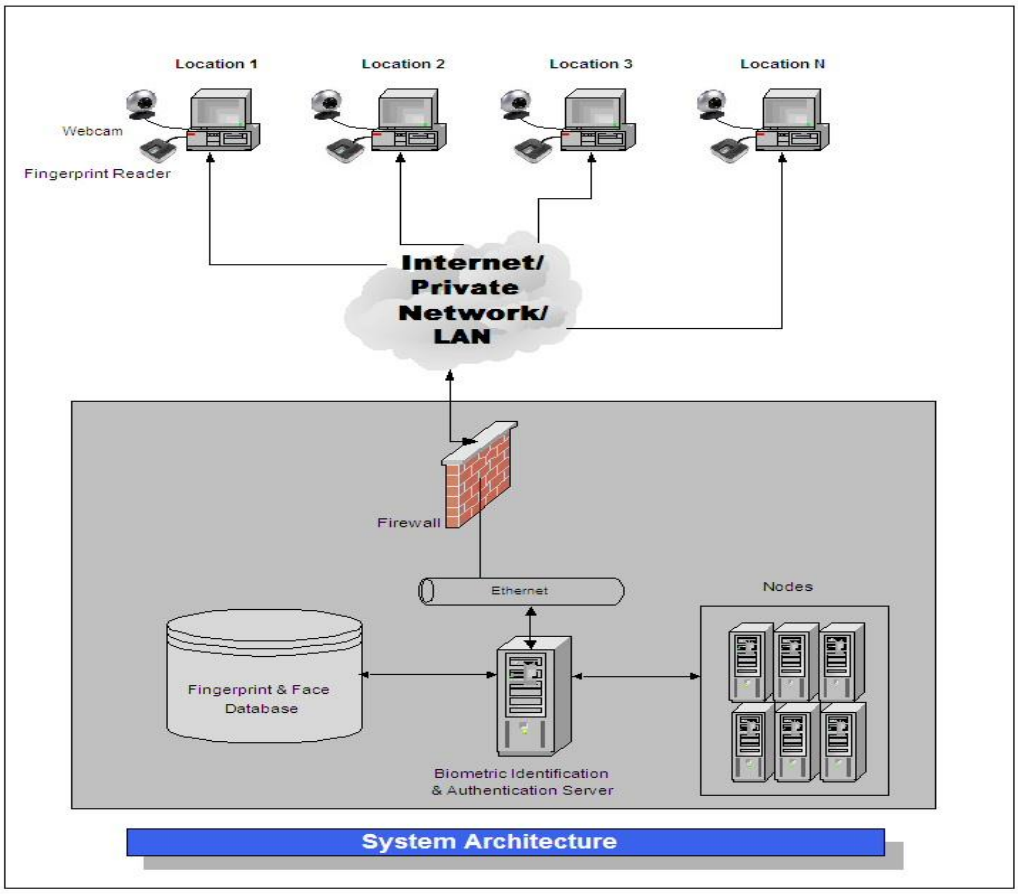


FIGURE 4: WEB-BASED BIOMETRICS SYSTEMS ARCHITECTURE [4]

1.3 MOTIVATION

When a large number of biometric records are present in the database it requires rapid and efficient searching method. With the increase in the size of the biometric database, reliability and scalability issues become the bottleneck for low response time, high search and retrieval efficiency in addition to accuracy [3]. Traditional identification systems claim identity of an individual by searching templates of all users enrolled in the database. These comparisons not only increase the data retrieval time but also the error rates. Hence a size reduction technique needs to be applied to reduce the search space and thus improve the efficiency. Conventionally databases are indexed numerically or alphabetically to increase the efficiency of retrieval. However, biometric databases do not possess a natural order of arrangement which negates the idea to index them alphabetically/numerically [6].

Certain *classification, clustering* and *indexing* approaches [7] [8] have been proposed to reduce search space. In supervised classification or discriminated analysis, a collection of labeled (pre-

classified) patterns are provided; the problem is to label a newly encountered, yet unlabeled, pattern. Typically, the given labeled (training) patterns are used to learn the descriptions of classes which in turn are used to label a new pattern. Several classification techniques exist like classification of face images based on age where input images can be classified into one of three age-groups: babies, adults, and senior adults. The main drawback of classification is that it is a supervised method where number of classes has to be known in advance. Moreover the data contained within each class is not uniformly distributed so the time required to search some classes is comparatively larger than some others. The limitations of classification can be addressed with unsupervised approach known as *Clustering*. It involves the task of dividing data points into homogeneous classes or clusters so that items in the same class are as similar as possible and items in different classes are as dissimilar as possible. Intuitively it can be visualized as a form of data compression, where a large number of samples are converted into a small number of representative prototypes [3].

Clustering can be broadly classified into Hard and Fuzzy clustering approaches. Non-fuzzy or hard clustering divides the data into crisp clusters, which is where each data point belongs to exactly one cluster. Fuzzy clustering in contrast, segments the data such that each sample data point can belong to more than one cluster and each data point has some 'degree of association' with every cluster. The sum of the membership grades of a particular data point belonging to more than one cluster is always one [2]. From the available biometric features collected by authors it can be inferred that each feature set has an association with more than one cluster and may have dissimilarity with data of the same cluster. In other words they are said to show inter class similarities and intra class variations, thus making them difficult to assign them to a single cluster. Hence fuzzy clustering techniques prove to be an efficient means for grouping biometric data. But the approach is not suitable for less number of clusters. However, as the size of the database increases the number of clusters required for partitioning also increases. The system using FCM has been tested earlier in [2] using bin-miss rate and performed better in comparison to traditional K-Means approach. But due to a higher bin-miss rate, it is not very accurate and so we go for an indexing technique for identification of large biometric database.

1.4 ORGANIZATION OF THE THESIS

The thesis has been organized in the following manner. An introduction to biometrics technology and its role in today's world has been presented in the first chapter of this thesis. The second chapter deals with an overview of the traditional clustering techniques. In the next chapter some of the tree data structures and the proposed indexing scheme have been discussed. Subsequently, the simulations and results have been shown, followed by the concluding remarks and the planned future work.

Chapter 2: LITERATURE STUDY PART I

2. AN OVERVIEW OF CLUSTERING METHODS

2.1 FUZZY C MEANS

Clustering involves arranging data points in such a way that the items sharing similar characteristics are grouped together. The goal of this process is to find the natural grouping of data points without prior knowledge of class labels (therefore it is unsupervised). Fuzzy C Means (FCM) is a feature clustering technique wherein each feature point belongs to a cluster by some degree that is specified by a membership grade [3]. These kind of clustering algorithms are known as objective function based clustering. If we are given *M* dimensional database of size *N* where *N* is the total number of feature vectors and *M* is the dimension of each feature vector, FCM assigns every feature vector a membership grade for each cluster. The problem here is to partition the database based on some fuzziness criteria using membership values. To find membership values, the partition matrix *U* of size $N \times c$ is calculated that defines membership degrees of each feature vector. The values 0 and 1 in *U* indicate no membership and full membership respectively. Grades between 0 and 1 indicate that the feature point has partial membership in a cluster. The algorithm is composed of the following steps:

1. *Initialize* $U=[u_{ij}]$ *matrix,* $U^{(0)}$
2. *At k-step: calculate the centers vectors* $C^{(k)}=[c_j]$ *with* $U^{(k)}$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} \cdot x_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

3. *Update* $U^{(k)}$, $U^{(k+1)}$

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

4. *If* $\| U^{(k+1)} - U^{(k)} \| < \varepsilon$ *then STOP; otherwise return to step 2.*

2.2 K MEANS

K-means is known to be one of the simplest unsupervised learning algorithms that can solve the well known clustering problem. A given data set is classified through the use of a certain number of clusters, let us assume k clusters and this number is fixed a priori. k centroids are defined, one for each cluster. Since different location causes different result, these centroids should be placed as much as possible far away from each other. In the next step each point belonging to a given data set is considered and associated to the nearest centroid. Proceeding this way, when no point is remaining, we can assume that the first step is completed and an early groupage is done. At this point, the k new centroids resulting from the previous step need to be re-calculated. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. In this manner, a loop has been generated. As a result of this loop the k centroids change their location step by step until the point when no more changes are done. In other words we stop when the centroids do not move anymore. This algorithm aims at minimizing an *objective function*, which is a squared error function in this case. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 ,$$

where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the *n* data points from their respective cluster centres. The algorithm is composed of the following steps:

1. *Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.*
2. *Assign each object to the group that has the closest centroid.*
3. *When all objects have been assigned, recalculate the positions of the K centroids.*
4. *Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.*

Here is the step-by-step k means clustering algorithm:

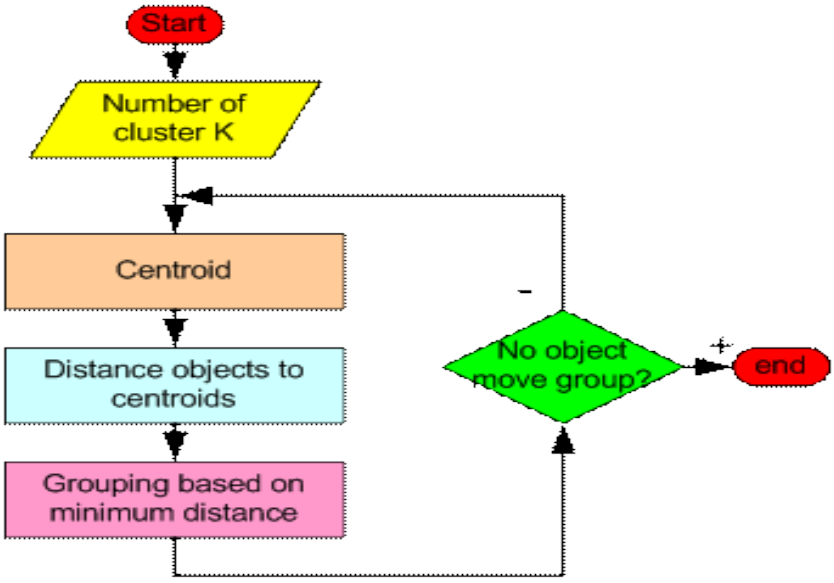


FIGURE 5: K-MEANS FLOWCHART

Step 1: Begin with a decision on the value of k = number of clusters.

Step 2: Put any initial partition that classifies the data into k clusters. The training samples may be assigned randomly, or systematically as the following:

1. Take the first k training samples as single-element clusters.
2. Assign each of the remaining (N-k) training samples to the cluster with the nearest centroid. After each assignment, re-compute the centroid of the gaining cluster.

Step 3: Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.

Step 4: Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

If the number of data is less than the number of clusters then we assign each data as the centroid of the cluster. Each centroid will have a cluster number. If the number of data is bigger than the number of clusters, for each data, we calculate the distance of all centroids and get the minimum distance. This data is said to belong to the cluster that has minimum distance from this data.

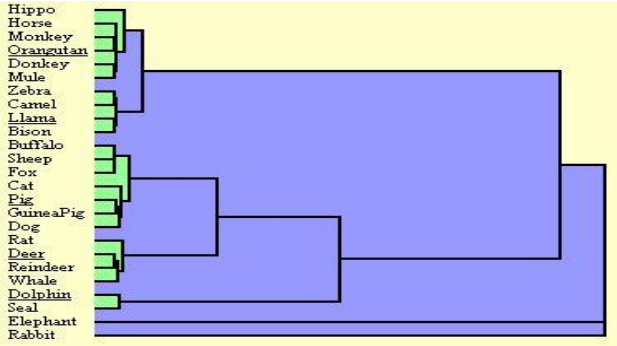


FIGURE 6: K-MEANS TREE [4]

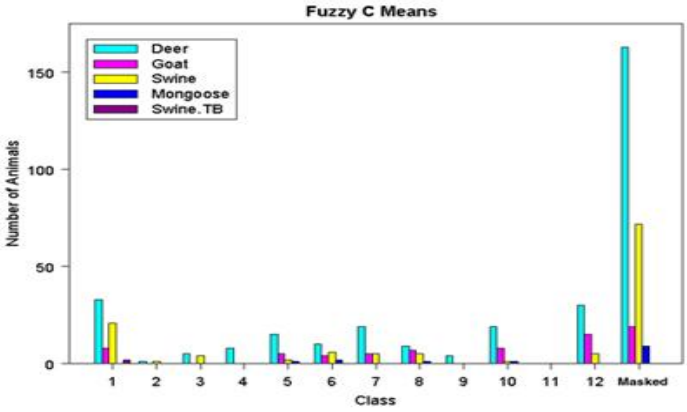


FIGURE 7: FUZZY C MEANS GRAPH [4]

Chapter 3: LITERATURE STUDY PART II

3. TREE DATA STRUCTURES AND INDEXING

3.1 BINARY SEARCH TREE [9]

Search trees are data structures that support many dynamic-set operations like SEARCH, MINIMUM, MAXIMUM, PREDECESSOR, SUCCESSOR, INSERT and DELETE. This work basically deals with search, insertion and deletion operations. Basic operations on a binary search tree (BST) take time proportional to the height of the tree.

WHAT IS A BST?

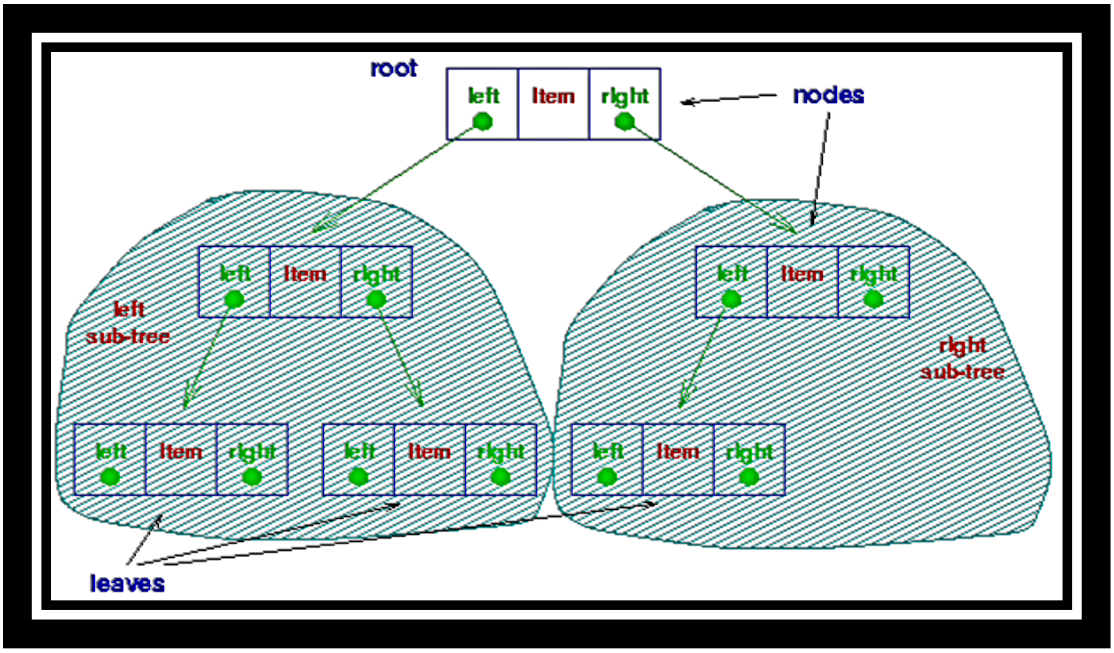


FIGURE 8: BINARY SEARCH TREE [4]

As the name suggests, a binary search tree is organized in a binary tree. It can be represented by a linked data structure in which each node is an object. In addition to a key field and satellite data, each node contains fields left, right and p that point to the nodes corresponding to its left child, its right child, and its parent, respectively. The keys in a binary search tree are always stored in such a way so that the binary-search-tree property is satisfied:

Considering x as a node in a binary search tree, if y is a node in the left subtree of x, then key[y] <= key[x], and if y is a node in the right subtree of x, then key[x] <= key[y].

On account of the binary-search-tree property we can print out all the keys in a BST in sorted order by a simple recursive algorithm, which is called an **inorder tree walk.** In this

algorithm, the key of the root of a subtree is printed between the values in its left subtree and those in its right subtree. Hence it is so named. In a similar manner, a **preorder tree walk** prints the root before the values in either subtree, and a **postorder tree walk** prints the root after the values in its subtrees.

INORDER-TREE-WALK (x)

```
if x != NIL
  then INORDER-TREE-WALK (left[x])
       print key[x]
       INORDER-TREE-WALK (right[x])
```

Similar are the algorithms for preorder and postorder tree walk.

Algorithms for SEARCH, MINIMUM, MAXIMUM, INSERT and DELETE are given.

Given a pointer to the root of the tree and a key k, TREE-SEARCH returns a pointer to a node with key k if one exists; otherwise, it returns NIL.

TREE-SEARCH (x, k)

```
if x = NIL or k = key[x]
then return x
if k < key[x]
then return TREE-SEARCH (left[x], k)
else return TREE-SEARCH (right[x], k)
```

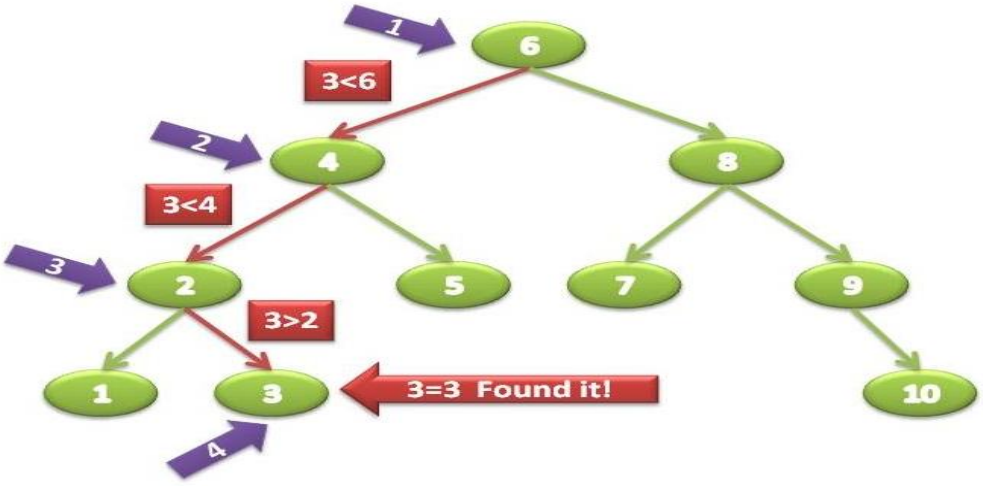


FIGURE 9: BINARY TREE SEARCH EXAMPLE [4]

An element in a binary search tree whose key is a minimum can always be found by following left child pointers from the root until a NIL is encountered. The procedure given below returns a pointer to the minimum element in the subtree rooted at a given node x.

TREE-MINIMUM (x)

```
while left[x] != NIL
  do x←left[x]
return x
```

The pseudo code for TREE-MAXIMUM is symmetric to that for TREE-MINIMUM:

TREE-MAXIMUM (x)

```
while right[x] != NIL
  do x←right[x]
return x
```

INSERTION

To insert a new value v into a binary search tree T, the procedure TREE-INSERT is used. The procedure is passed a node k for which key[k] = v, left[k] = NIL, and right[k] = NIL.

TREE-INSERT (T, k)

```
y←NIL
x←root[T]
while x != NIL
  do y←x
if key[k] < key[x]
  then x←left[x]
  else x←right[x]
p[k]←y
if y = NIL
  then root[T]←k  //  Tree T was empty
  else if key[k] < key[y]
  then left[y]←k
  else right[y]←k
```

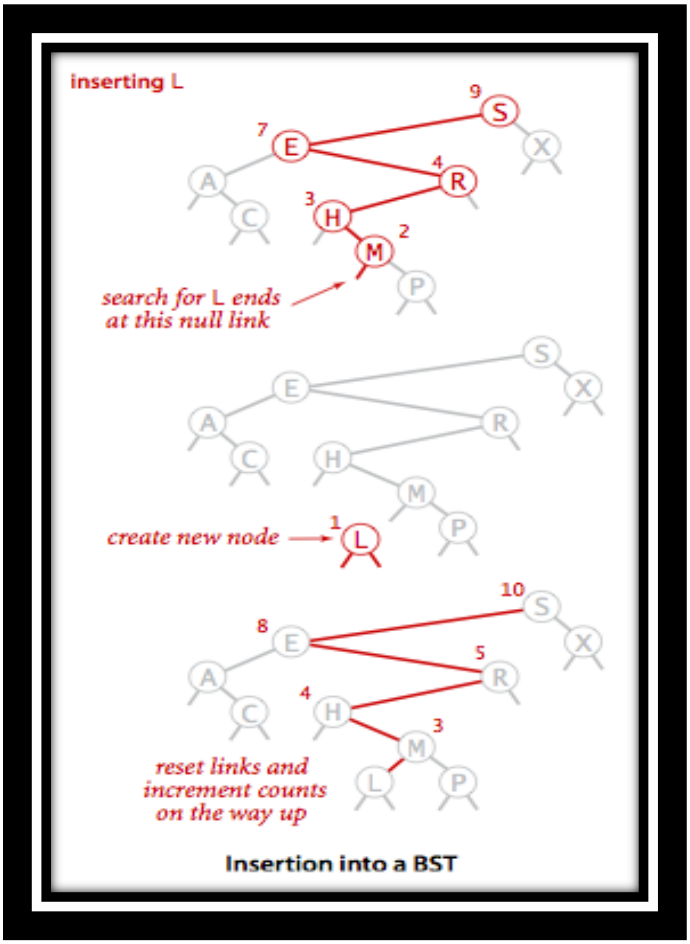


FIGURE 10: INSERTION INTO A BINARY SEARCH TREE [4]

DELETION

The procedure for deleting a given node k from a BST takes as an argument a pointer to k. The procedure considers three cases: a) if k has no children, we modify its parent p[k] to replace k with NIL as its child. b) if the node has only a single child, we "splice out" k by making a new link between its child and its parent. c) finally, if the node has two children , we splice out k's successor y, which has no left child and replace k's key and satellite data with y's key and satellite data.

TREE-DELETE (T, k)

if left[k] = NIL or right[k] = NIL

then y ← k

else y ← TREE-SUCCESSOR (k)

```
if left[y] != NIL
      then x ← left[y]
      else x ← right[y]
if x != NIL
      then p[x] ← p[y]
if p[y] = NIL
     then root[T] ← x
     else if y = left[p[y]]
               then left[p[y]] ← x
               else right[p[y]] ← x
if y != k
   then key[k] ← key[y]
        copy y's satellite data into k
return y
```

3.2 FEW SALIENT FEATURES OF TREES

3.2.1 BINARY TREES, B-TREES AND B+-TREES

Binary tree is a tree data structure in which each node has at most two children. Typically the first node is known as the parent and the child nodes are called left and right.

The B-tree is a generalization of a binary search tree in that more than two paths diverge from a single node. This tree data structure that keeps data sorted and allows searches, insertions, deletions, and sequential access in logarithmic amortized time. Unlike *self-balancing binary search trees*, the B-tree is optimized for systems that read and write large blocks of data and is most commonly used in databases and file systems. It is a specialized multi way tree which is especially designed for use on disk. In a B-tree, each node may contain a large number of keys. The number of sub trees of each node, then, may also be large. A B-tree is designed to branch out in this large number of directions and to contain a lot of keys in each node as a result of

which the height of the tree is relatively small. This means that only a small number of nodes must be read from disk in order to retrieve an item. The goal is to get fast access to the data, and with disk drives this means reading a very small number of records.

The most attractive feature of B-tree is its small memory usage. A binary tree needs at least two pointers for each record, which amounts to 16N on a modern 64-bit system. A B-tree only needs one pointer. The practical memory overhead can be reduced as the majority of nodes in a B-tree are leaves, the performance is supposed to be far better than a binary search tree. Whereas some others have the view that they are totally different in their uses, hence no comparison between them can be made at all [3].

DEFINITION OF B TREES [7]

A B tree T which is a rooted tree has the following properties:

1. Every node x contains the following fields:
   - n[x] , which is the number of keys currently stored in node x,
   - the n[x] keys themselves stored in such an order that $key_1[x] <= key_2[x] <= \ldots <= key_{n[x]}[x]$,
   - leaf [x], which is a Boolean value that is TRUE if x is a leaf and FALSE if x is an internal node.

2. Every internal node also has n[x] + 1 pointers $c_1[x], c_2[x], \ldots, c_{n[x]+1}[x]$ to its children. Since leaf nodes have no children, so their c_i fields are not defined.

3. The keys $key_i[x]$ separate the ranges of keys stored in each subtree, that is to say, if k_i is any key stored in the subtree with root $c_i[x]$, then

$$k_1 <= key_1[x] <= key_2[x] <= \ldots <= key_{n[x]}[x] <= k_{n[x]+1}.$$

4. All leaves have the same depth, which is equal to the height of the tree 'h'.

5. There are upper and lower bounds on the number of keys a node can contain, which can be expressed in terms of a fixed integer t >= 2 and it is called the 'minimum degree' of the B tree:

   a. Every node other than the root must have at least (t-1) keys
   b. Every node can contain at most (2t-1) keys.
      [We get the simplest B tree when t = 2. In this case every internal node has either 2, 3, or 4 children, and we call it a **'2-3-4' tree**.]

B trees generalize binary search trees in a natural manner: if an internal B tree node x contains n[x] keys, then x has (n[x] + 1) children. The keys in node x are used as 'dividing points' which separate the range of keys handled by x into (n[x] + 1) sub ranges, each of which is handled by

one child of x. **Searching** a B tree is much similar to searching a binary search tree, except that instead of making a binary, or "two-way", branching decision at each node, a "multi way branching decision" is made according to the number of the node's children. To be more precise, at each internal node x, an (n[x] + 1)-way branching decision is made.

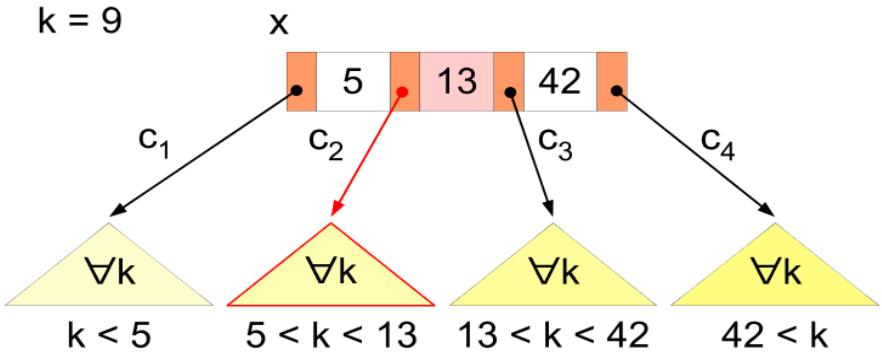


FIGURE 11: B TREE SEARCH [3]

Inserting a key into a B tree is significantly more complicated than inserting a key into a binary search tree. As with BST, we search for the leaf position at which to insert the new key. With a B tree we cannot simple create a new leaf node and insert it. Then the resulting tree would not be a valid B tree. Instead, we insert the new key into an existing leaf node. Since we cannot insert a key into a leaf node that is "full", we use an operation here that "splits" a full node y (having 2t – 1 keys) around its median key $key_t[y]$ into two nodes having (t – 1) keys each. The median key moves up into y's parent to identify the "dividing point" between the two new trees. But if y's parent is also full, it must be split again before inserting the new key, and this may continue all the way up the tree.

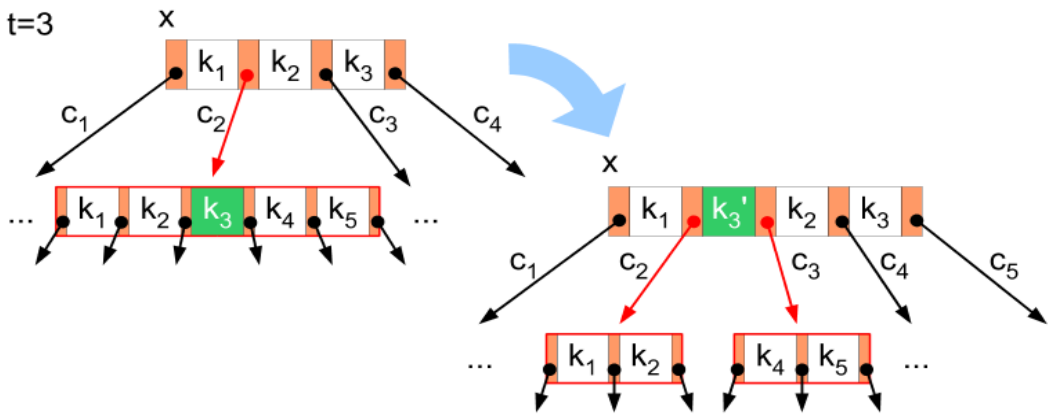


FIGURE 12: B TREE SPLIT [3]

3.2.2 DEFINITIONS RELATED TO B+ TREE [10]

B+ tree

- is a structure of nodes linked by pointers
- is anchored by a special node called the root, bounded by leaves
- has a unique path to each leaf, and all paths are of equal length
- stores keys only at leaves, and stores reference values in other internal nodes
- guides key search, via the reference values, from the root to the leaves

Node

- is either internal, or leaf, including the root node
- contains at most n entries and one extra pointer for some fixed n
- has no fewer than $\lfloor n/2 \rfloor$ entries, the root excepted

Root node

- is a leaf when it is the only node in the tree and will then contain at least one entry
- must have at least two pointers to other nodes when it is internal

Internal node

- contains entries consisting of a reference value and a pointer towards the leaves
- its entries point to data classified as greater than or equal to the corresponding reference value
- its extra pointer references data classified as less than the node's smallest reference value

Leaf node

- contains entries consisting of a key value and a pointer to the storage location of data matching the key
- its extra pointer references the next leaf node in the tree ordering; leaves linked in this manner are neighbors

A **B+ tree** is a type of tree which represents sorted data in a way that allows for efficient insertion, retrieval and removal of records, each of which is identified by a *key*. It is a dynamic, multilevel index, with maximum and minimum bounds on the number of keys in each index segment (usually called a "block" or "node"). In a B+ tree, in contrast to a B-tree, all records are stored at the leaf level of the tree; only keys are stored in interior nodes. The primary value of a B+ tree is in storing data for efficient retrieval in a block-oriented storage context. This is primarily because unlike binary search trees, B+ trees have very high fanout (typically on the

order of 100 or more), which reduces the number of I/O operations required to find an element in the tree.

In B-tree structures, key search proceeds from the root downwards, following pointers to the nodes which contain the appropriate range of keys, as indicated by the reference values. Likewise, B-trees grow from the leaves upwards. After obtaining the appropriate location for the new entry, it is inserted into the tree. If the node becomes overfull, it is split into half and a pointer to the new half is returned for insertion in the parent node, which if also full will in turn split again and so on. B+-trees distinguish internal and leaf nodes, keeping data only at the leaves, whereas ordinary B-trees would also store keys in the interior.

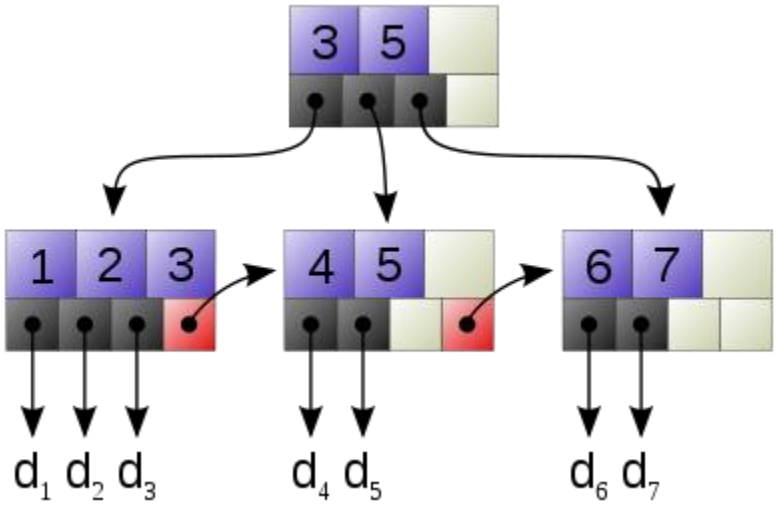


FIGURE 13: B+ TREE EXAMPLE [4]

There are two types of nodes that a B+ tree typically consists of viz. internal nodes and leaf nodes. Every internal node points to another node in the tree while leaf node points to the IDs in the database through data pointers. Leaf nodes also contain another pointer called sibling pointer, which points to the next leaf node. The figure below shows the structure of an internal node of a B+ tree of order p where K_1, K_2, … K_q are keys satisfying $K_1<K_2<\ldots K_{q-1}$, $q<=p$ and P_i points to a subtree S containing all key values more than K_{i-1} but less than or equal to K_i.

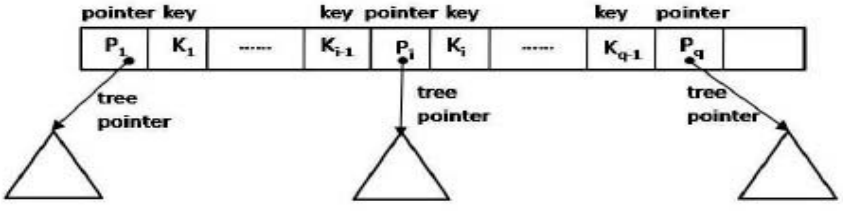


FIGURE 14: INTERNAL NODE OF B+ TREE OF ORDER p [1]

The following figure depicts the structure of a leaf node where D_i is a data pointer pointing to an ID having key feature value K_i satisfying $K_1<K_2<\ldots K_p$. All leaf nodes are at the same level. The number of disk accesses required for most operations on a B+ tree is proportional to the height of the B+ tree [1].

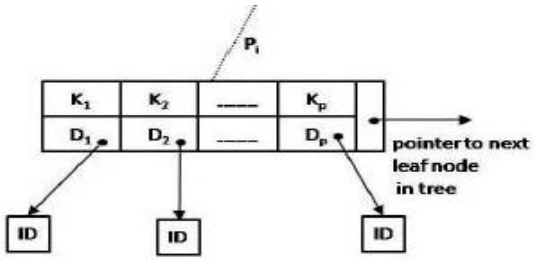


FIGURE 15: LEAF NODE OF B+ TREE OF ORDER p [1]

3.3 MODIFIED B+ TREE

Some variation has to be done in the original B+ tree structure n order to meet the requirements of insertion and searching in biometric data. In a modified B+ tree of order p, having n feature values and N IDs, instead of storing a single feature value as key, range R_i on feature values is to be computed and stored as range. The structure of an internal node of the modified B+ tree of order p is clear from the figure shown. Here R_1, R_2, … R_q are range satisfying $R_1<R_2<\ldots R_{q-1}$, q<=p. (All other properties remain same as B+ tree.)

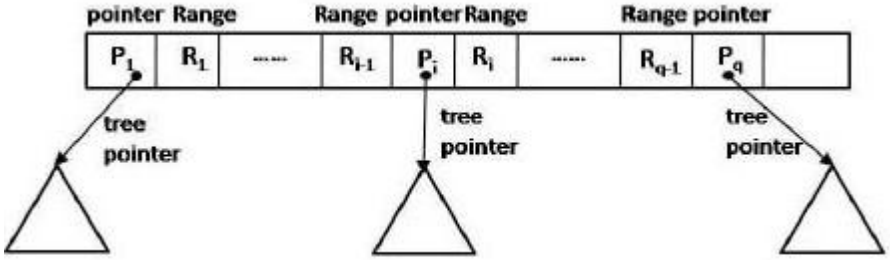


FIGURE 16: INTERNAL NODE OF MODIFIED B+ TREE OF ORDER p [1]

The next figure illustrates the structure of a leaf node where D_i is the multiple data pointer pointing to a set of IDs having range feature values R_i satisfying $R_1<R_2<\ldots R_p$. (All other properties remain same). The advantage of this modification is that it reduces the height of the tree.

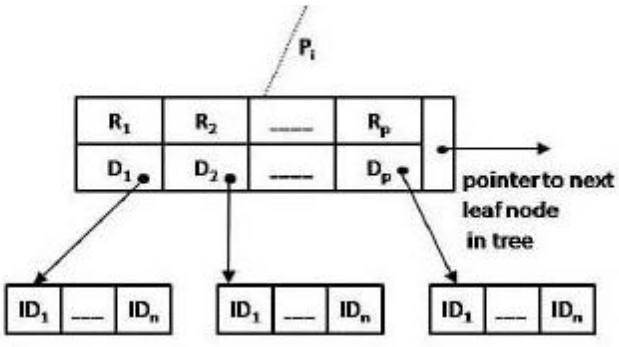


FIGURE 17: LEAF NODE OF MODIFIED B+ TREE OF ORDER p [1]

The Indexing Technique Used

In [2], N feature vectors $F_1,F_2,\ldots F_N$ have been considered. Each feature vector of m dimension has been defined as follows.

$$F_1 = [f_{11},\ f_{12} \ldots\ldots f_{1m}]$$

$$F_2 = [f_{21},\ f_{22} \ldots\ldots f_{2m}]$$

$$\cdot$$

$$\cdot$$

$$F_N = [f_{N1},\ f_{N2} \ldots\ldots f_{Nm}]$$

After that feature vectors FC_i, i = 1,2,…m have been defined where FC_i consists of all i^{th} feature values of $F_1,F_2,\ldots F_N$ as follows:

$$FC_i = [f_{1i},\ f_{2i} \ldots\ldots f_{Ni}]$$

In the technique proposed in [2], these feature vectors $FC_1,FC_2,\ldots FC_m$ are considered as the keys for indexing. The generalized structure of modified B+ tree is shown in the following figure where FE is the feature value as range R_i and IDs are the identifiers as multiple data pointers.

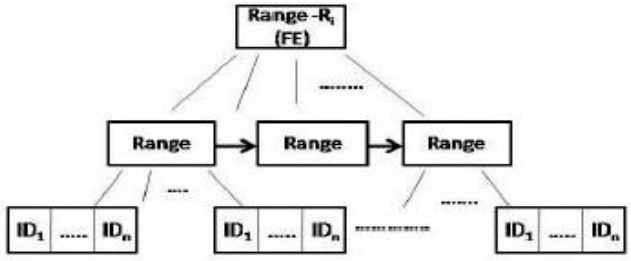


FIGURE 18:THE GENERALIZED STRUCTURE OF MODIFIED B+ TREE [1]

An example has been considered to determine the modified B+ tree. Table 1 contains 3 feature values f_1, f_2, f_3 for 10 IDs. Subsequently, the corresponding B+ tree and modified B+ tree for the feature value f_2 of order 2 have been shown.

TABLE 1: FEATURE VALUES [2]

	f_1	f_2	f_3
ID_1	4.677	2.005	5.0823
ID_2	3.455	5.555	3.755
ID_3	2.455	4.544	1.825
ID_4	5.666	4.999	4.154
ID_5	2.345	3.999	2.970
ID_6	4.009	5.913	5.695
ID_7	5.002	3.738	2.816
ID_8	2.234	2.388	3.835
ID_9	1.345	2.950	5.719
ID_{10}	1.988	5.614	4.708

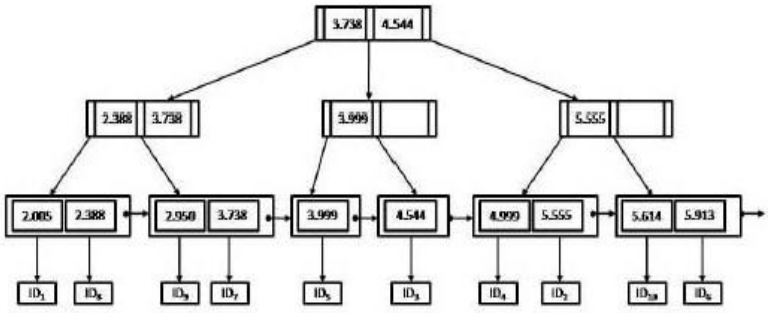


FIGURE 19: B+ TREE FOR FEATURE VALUE f2 OF ORDER 2 [1]

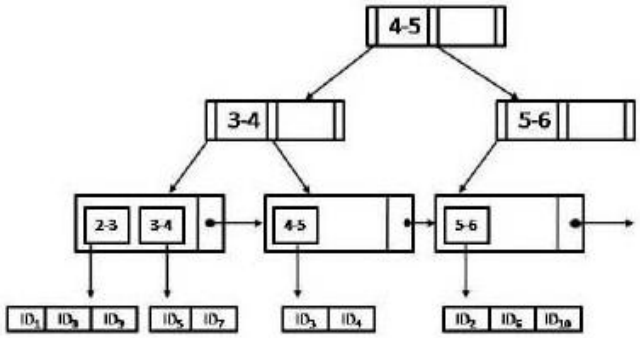


FIGURE 20: MODIFIED B+ TREE FOR FEATURE VALUE f2 OF ORDER 2 [1]

3.3.1 INSERTION IN MODIFIED B+ TREE [1]

Algorithm 1 INSERT (feature value v) : In Modified B+ Tree

1. Compute range R_i for the feature value v.
2. Determine the node containing the range R_i.
3. if the range node R_i is found then
4. Insert only ID of v in the range node.
5. else
6. Create a node for the range R_i, insert ID of v.
7. end if

The insertion procedure to insert a key in the Modified B+ tree is similar to that in the B+ tree. For every feature value of FC_1,FC_2,…FC_m a Modified B+ tree T is formed. For a given feature value, a range R_i is computed. Then, by traversing the tree, an appropriate range node is found in which the new value lies. Once this range has been found, the ID of the feature value is inserted at the corresponding leaf node. (Note: In the Modified B+ tree for the feature vectors FC_1,FC_2,…FC_m, it inserts the range as key, instead of the feature value itself.)

3.3.2 SEARCHING IN MODIFIED B+ TREE [1]

Algorithm 2 SEARCH (feature value q) : In Modified B+ tree

1. Compute range R_i for the query feature value q.
2. R_i be the input search range and RANGE be the range stored in the nodes.
3. Start the searching at the root.

4. if if we encounter an internal node v then
5. search for R_i among the RANGE stored at v.
6. if $R_i < RANGE_{min}$ at v then
7. follow the left child pointer.
8. end if
9. if $RANGE_i <= R_i < RANGE_{i+1}$ for two consecutive $RANGE_i$ and $RANGE_{i+1}$ at v then
10. follow the left child pointer of $RANGE_{i+1}$.
11. end if
12. if $R_i >= RANGE_{max}$ at v then
13. follow the right pointer of $RANGE_{max}$.
14. end if
15. end if
16. if if we encounter a leaf node *l* then
17. retrieve all IDs from the node *l* stored RANGE.
18. end if

The problem is as follows: given a Modified B+ tree consisting of n range values as a key, whose range value is feature values of N feature vectors, we have to search the smallest range in which a given query vector Q of m dimension $[q_1,q_2,\ldots q_m]$ lies. For every feature value q_i of the query template Q, the range R_i is to be computed. By traversing the tree we reach to the leaf node having the range R_i. It is then that a set of IDs is extracted which is nearest possible match corresponding to q_i.

PROBLEM STATEMENT

Given a query image, suppose Q, the problem as defined is to reduce the search space in the database consisting of N individuals say ID_1, ID_2, ID_3, … , ID_N, each having a unique ID. We consider that the biometric trait generates feature vector F and it consists of m feature values for an individual ID. Let $f_{i,j}$ be the feature value in the j^{th} dimension for all IDs i; all feature values $f_{i,j}$ are lying between 'a' and 'b' where a and b are defined as

$$a = \min_{i \in N}\{f_{ij}\} \text{ for a given } j$$

$$b = \max_{i \in N}\{f_{ij}\} \text{for a given } j$$

Since the biometric system uses pattern recognition technique, it is most unlikely that there is an image in the given database which has exactly the same feature values as those of query image. In other words, if the query image Q consists of feature values of m dimension defined as Q = $[q_1, q_2, \ldots q_m]$ then for all j, q_j may not be same as $f_{i,j}$, where q_j is the j^{th} feature value of Q. The

feature values of each individual can be arranged in such a way that an efficient searching algorithm can be used, since these values are known a priori. A possible data structure that can be used for this purpose is a Binary Search tree or a B tree or a B+ tree whose key values are these feature values.

Prior to using Modified B+ tree for indexing, this work implements indexing using a Binary Search tree by reading values stored in a database. The same has been implemented for B tree and their complexities have been compared.

3.4 RED-BLACK TREES

A **red-black tree** is one of self-balancing binary search trees, a data structure primarily used to implement associative arrays. The original structure was invented in 1972 by Rudolf Bayer who called them "symmetric binary B-trees", but acquired its modern name in a paper in 1978 by Leonidas J. Guibas and Robert Sedgewick. This data structure is complex, but at the same time has good worst-case running time for its operations and is efficient in practice that is, it can search, insert, and delete in O(log *n*) time, where *n* is total number of elements in the tree [4]. Putting in simple words, a red-black tree is a binary search tree which inserts and removes 'intelligently', to ensure the tree is 'reasonably balanced'.

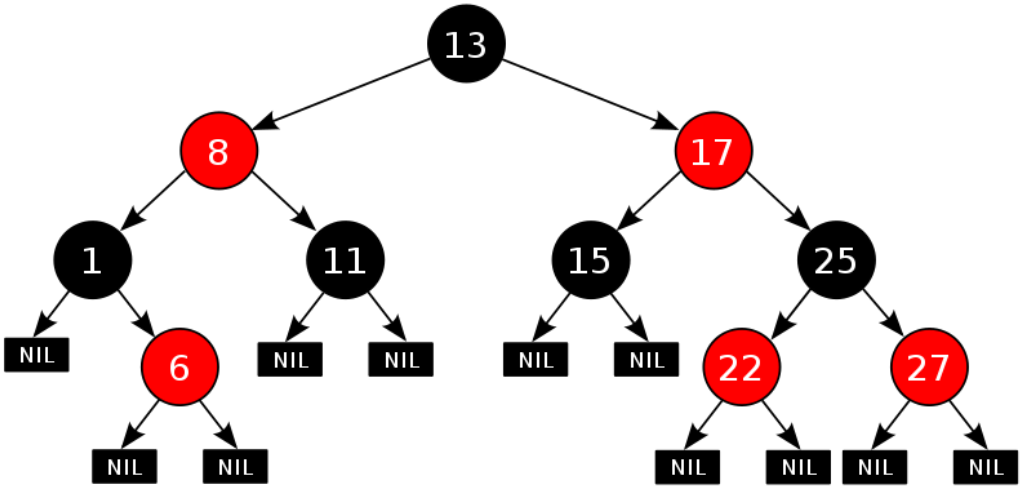


FIGURE 21 : AN EXAMPLE OF A RED-BLACK TREE [4]

This special type of binary tree is used in computer science to organize pieces of comparable data, such as text fragments or numbers. The leaf nodes do not contain data and they need not be explicit in computer memory. To save memory, sometimes a single sentinel

node performs the role of all leaf nodes; where all references from internal nodes to leaf nodes point to the sentinel node. Red-black trees, like all binary search trees, allow efficient in-order traversal of their elements in Left-Root-Right fashion. The search-time results from the traversal from root to leaf, and therefore a balanced tree, having the least possible tree height, results in O(log *n*) search time.

3.4.1 PROPERTIES OF RED-BLACK TREES

A red-black tree is a binary search tree with one extra bit of storage per node --- its color that is, each node has a *color* attribute, the value of which is either *red* or *black*. Then, each node of the tree contains the fields *color*, *key*, *left*, *right*, and *p*. If a child or the parent of a node does not exist, the corresponding pointer field of the node contains the value NIL. In addition to the ordinary requirements imposed on binary search trees, the following additional requirements apply to red-black trees [9]:

1. A node is either red or black.
2. The root is black.
3. All leaves are black.
4. Both children of every red node are black.
5. Every simple path from a given node to any of its descendant leaves contains the same number of black nodes.

These constraints enforce a critical property of red-black trees --- that the longest path from the root to any leaf is no more than twice as long as the shortest path from the root to any other leaf in that tree. As a result of this property, the tree is roughly balanced. Operations such as inserting, deleting, and finding values require worst-case time proportional to the height of the tree. Therefore, unlike ordinary binary search trees, this theoretical upper bound on the height allows red-black trees to be efficient in the worst-case.

3.4.2 OPERATIONS ON A RED-BLACK TREE

Since every red-black tree is a special case of a simple binary search tree, the read-only operations on a red-black tree require no modification from those used for binary search trees. However, the immediate consequence of an insertion or a removal on a red-black tree may violate the properties of a red-black tree. Restoring the red-black properties requires a small number of color changes which is O(log *n*) or amortized O(1) and no more than three tree rotations (two in case of insertion). Although insert and delete operations are complicated yet their times remain O(log *n*).

ROTATION

When run on a red-black tree with n keys, the search-tree operations TREE-INSERT and TREE-DELETE take O (lg n) time. Because they modify the tree, the result of these operations may violate the red-black properties, which has been enumerated earlier. To restore these properties, the colors of some of the nodes in the tree must be changed and we also need to change the pointer structure. The pointer structure can be changed through *rotation*, which is a local operation in a search tree that preserves the binary-search-tree property. There are two kinds of rotations – 1. left rotations, and 2. right rotations. When we do a left rotation on a node x, it is assumed that its right child y is not nil[T]. The left rotation "pivots" around the link from x to y. y is made the new root of the subtree, with x as y's left child and y's left child as x's right child [9]. Similar to this is the right rotation. Both procedures run in O(1) time.

INSERTION

Insertion in a red-black tree begins by adding the node much as binary search tree insertion does in addition, only by coloring it red. In the binary search tree a leaf node is always added, whereas in the red-black tree as we have seen, leaves contain no information, so instead a red interior node with two black leaves is added in place of an existing black leaf.

What happens next depends on the color of other neighboring nodes. The term *uncle node* is used to refer to the sibling of a node's parent, just like in human family trees [4].

- Property 3 which states that all leaves are black always holds.
- Property 4 stating that both children of every red node are black is threatened either by adding a red node, repainting a black node red, or a rotation.
- Property 5, according to which all paths from any given node to its leaf nodes contain the same number of black nodes, is threatened only by adding a black node, repainting a red node black, or a rotation.

REMOVAL

In the case of a normal binary search tree, when deleting a node with two non-leaf children, we find either the maximum element in its left subtree or the minimum element in its right subtree, and move its value into the node being deleted. Then the node we copied the value from is deleted, which must have less than two non-leaf children. This reduces to the problem of deleting a node with at most one non-leaf child because merely copying a value does not violate any red-black properties. Whether this node is the node we originally wanted to delete or the node we copied the value from does not really matter.

If a red node is to be deleted, we can simply replace it with its child, which must be black (as we already know, a red node can have either two non leaf black children or two leaf children which are black as per definition, thus in this case the red node is replaced by a leaf because it was

required the node to be deleted has at most one non leaf child). Then, all paths through the deleted node will simply pass through one less red node, and both the deleted node's parent and child must be black. As can be inferred, properties 3 (All leaves, including *null*s, are black) and 4 (Both children of every red node are black) still hold.

We can consider another simple case as when the deleted node is black and its child is red. If we simply remove a black node could break Properties 4 (Both children of every red node are black) and 5 (All paths from any given node to its leaf nodes contain the same number of black nodes). But if we repaint its child black, both of these properties are preserved.

3.5 HOW DOES A DATABASE INDEX WORK?

An index

- is generally sorted by key values that need not be the same as those of the table.
- is small and has just a few columns of the table.
- refers to the right block within the table for a key value.
- speeds up reading a row, provided one knows the right search arguments.

A **database index** is a data structure that improves the speed of data retrieval operations on a database table but at the cost of slower writes and increased storage space. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random look ups and efficient access of ordered records. Since indexes usually contain only the key-fields according to which the table is to be arranged, and excludes all the other details in the table, therefore the disk space required to store the index is typically less than that required by the table. This yields the possibility to store indexes in memory for a table whose data is too large to store in memory.

WHAT IS INDEXING AND WHY IS IT NEEDED?

Indexing is a way of sorting a number of records on multiple fields. Creating an index on a field in a table creates another data structure which holds the field value, and pointer to the record it relates to. This index structure is then sorted, allowing Binary Searches to be performed on it. When data is stored on disk based storage devices, it is stored as blocks of data. These blocks are accessed in their entirety, making them the atomic disk access operation. Disk blocks are structured in much the same way as linked lists; both contain a section for data, a pointer to the location of the next node (or block), and both need not be stored contiguously [3].

Chapter 4

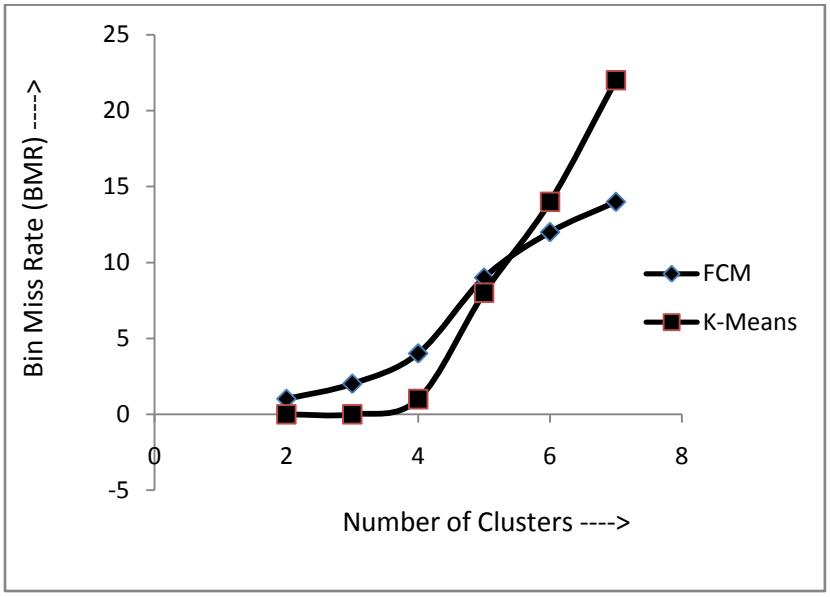
4. SIMULATION AND RESULTS

4.1 K-MEANS AND FCM CLUSTERING ALGORITHMS

The k-means algorithm was implemented on a sample data set using C language on Windows platform. The initial data set consisted of around 8-10 values on which the K-Means clustering algorithm has been applied taking the value of k (number of clusters) as 2. Then the same has been implemented on a larger database consisting feature values from 500 individuals with the value of k ranging from 2 to 7. The system has also been tested using the FCM clustering algorithm. The results obtained are given in the table below.

TABLE 2: FCM VS K-MEANS

Number of Clusters	FCM	K-Means
2	1	0
3	2	0
4	4	1
5	9	8
6	12	14
7	14	22



GRAPH 1: COMPARISON OF FCM AND K-MEANS

This project work is a continuation of the work [3] in which a new identification strategy by partitioning a biometric database using *clustering* has been proposed. In the previous work, the fuzziness criterion has been introduced for finding the nearest clusters for declaring the identity of the query sample. As had been obtained in [2], similar results have been obtained on implementing the K-Means clustering algorithm and Fuzzy C Means (FCM) clustering algorithm in the present work. It has been observed that for less number of clusters the K-Means approach works comparatively better than the FCM approach. But as the size of database increases, the number of clusters required for partitioning also increases. With an increase in the number of clusters, there is a higher bin-miss rate in K-Means and FCM performs comparatively better though does not provide very accurate results with more number of clusters. From the Graph 1 and the Table 2, we also infer:

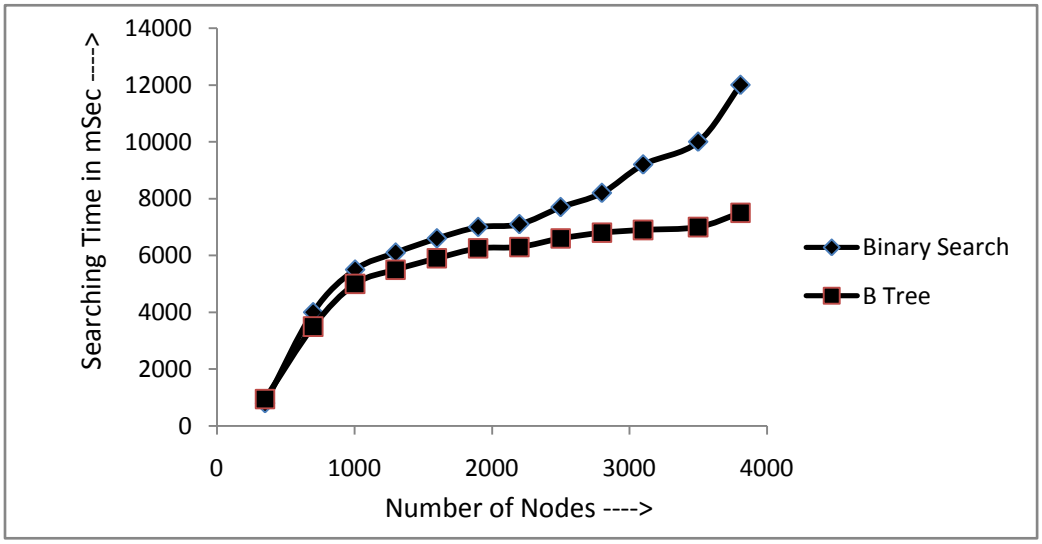
a. Clustering approaches like K-Means and FCM perform well with less number of partitions.
b. Clustering biometric database gives higher bin-miss rate.
c. There is a need to develop robust identification strategy using some indexing techniques like tree data structures or hashing.

4.2 INDEXING USING BINARY SEARCH TREE AND B TREE

The basic objective of the next work is implementation of the paper [2] which deals with an efficient indexing technique using Modified B+ tree to reduce the search space of large biometric database. An indexing method helps to declare a person's identity with lesser number of comparisons rather than searching the entire database. Before proceeding to B+ tree, the same has been implemented using a Binary Search Tree (BST) using Java language with NetBeans 6.0 IDE tools on Windows platform. Initially, a simple binary search tree has been implemented which takes as input some well-defined range of values (integer as well as float). The next job was to perform the same on a given database. For that purpose, a file has been created and stored consisting of a small sample biometric database. The feature values from this sample database have been read and inserted into the tree and then indexing has been performed on the database using this binary search tree. After the values have been inserted and the tree has been built, searching and removal of a node (element) has been done successfully. Next, implementation of indexing using B tree has been done on the same data set. Graphs have been plotted indicating the performance of these two trees. The results of the two simulations have been compared. The tree operations search, insertion and deletion have been implemented both for Binary Search tree and B tree. Before implementing these basic dynamic set operations, the preorder, inorder and postorder traversals of BST have been implemented using the Java language. A random set of integers was generated and inserted into the tree and then the traversals have been performed in order to test that the tree is working properly.

TABLE 3 : VALUES OF NO. OF NODES AND THEIR RESPECTIVE TIMES OF BUILDING BST AND B TREE

Number of Nodes	Binary Search	B Tree
350	800	950
700	4000	3500
1005	5500	5000
1300	6100	5500
1600	6600	5900
1900	7000	6250
2200	7100	6300
2500	7700	6600
2800	8200	6800
3100	9200	6900
3500	10000	7000
3808	12000	7500



GRAPH 2: GRAPH SHOWING NO. OF NODES VS TIME REQUIRED IN MSEC

4.3 CONCLUSION AND DISCUSSION

As can be observed from Graph 1 and Table 2, K-Means clustering technique performs better when the size of the database and hence the number of clusters is less. On the other hand, FCM gives higher bin-miss rate comparatively. But with increase in the size of the database, naturally the required number of partitions also increases. With this, K-Means has a higher bin-miss rate as compared to FCM but the results obtained by both the methods are not very accurate and satisfactory. So in order to improve accuracy as well as speed of data retrieval from the database, indexing schemes using Binary Search Tree and B Tree have been applied on the sample database.

The primary operations on a binary search tree require time proportional to the height of the tree, i.e. time O(h) if the binary search tree is of height h. The insertion operation, in the worst case, takes time proportional to the height of the tree, whereas it is O(log *n*) time in the average case. The search operation also requires O(log *n*) time in the average case, but in the worst-case needs O(*n*) time [9]. As can be observed from GRAPH 2, the time required (in mSec) to create a binary search tree and insert into it the values read from the database is proportional to the logarithm of number of nodes in the tree to be built. In other words, given 'n' to be the number of nodes to be inserted into the tree, the procedure can be implemented in O(log n) time (time complexity). Similarly searching a node and deletion of a node also take O(log n) time. This proves indexing a biometric database and then searching it or performing other operations on it would be faster and possibly less error-prone as compared to clustering. But the graph obtained is not very smooth and accurate, thus there is scope for improvement. So, the same has been implemented using B tree.

After the B-Tree has been implemented, a graph has been plotted with the results obtained and from these two graphs a comparison has been drawn between the two types of trees used. From the graph of binary search tree, the time is not exactly (but roughly) proportional to log (n). The graph of B tree is almost similar but as derived from the execution (run) time of the application, has time nearly proportional to log (n), i.e. with more number of nodes it takes lesser time to run and thus performs comparatively superior. Though from this we cannot directly conclude that B trees always necessarily perform better as compared to binary search trees. Indexing techniques using Binary Search Tree and B Tree have yielded similar and better though not perfect results. Implementing B tree indexing has not proved to be much of an improvement over indexing using Binary search trees. Indexing the database using B+ tree and Modified B+ tree was the ultimate objective of this project, but due to the timing constraints it could not be implemented. Future work is planned to be performed on it. A study of the Red-Black trees has been done and since it is known that it has good worst-case running time for its operations and is efficient: it can search, insert, and delete in O(log *n*) time, *n* being total number of elements in the tree, so further work can also be based on implementation of Red-Black trees to improve space and time complexity.

BIBLIOGRAPHY

[1] Arun Ross, Salil Prabhakar Anil K. Jain, "An Introduction to Biometric Recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. VOL. 14, No. 1, pp. 4-20, January 2004.

[2] Surya Prakash, Devdatt and Phalguni Gupta U. Jayaraman, "An Indexing Technique for Biometric Database," in *IEEE*, Aug, 2008, pp. 758-761.

[3] V. Bhawani Radhika, "Biometric Identification Systems: Feature Level Clustering of Large Biometric Database," NIT Rourkela, May,2009.

[4] Wikipedia. [Online]. http://en.wikipedia.org

[5] Patrick Flynn, Arun A. Ross Anil K. Jain, "Introduction to Biometrics," in *Handbook of Biometrics*.: Springer US, 2007, pp. 1-22.

[6] Dakshina R. Kisku, V. Bhawani Radhika, B. Majhi, Phalguni Gupta Hunny Mehrotra, "Feature Level Clustering of Large Biometric Database," pp. 324-326, May, 2009.

[7] Badrinath G. Srinivas, Banshidhar Majhi and Phalguni Gupta Hunny Mehrotra, "Indexing Iris Biometric Database Using Energy Histogram of DCT subbands," pp. pp. 194–204, 2009.

[8] Sharat Chikkerur and Venu Govindaraju Amit Mhatre, "Indexing Biometric Databases using Pyramid Technique," *Lecture Notes in Computer Science*, vol. 3546, pp. 841-849, 2005.

[9] Charles E. Leiserson, Ronald L. Rivest, Clifford Stein Thomas H. Cormen, *Introduction to Algorithms*.: PHI.

[10] Jan Jannink, "Implementing Deletion in B+-Trees," pp. 33-34, March, 1995.