

Volume 7 ■ Issue 2 ■ September 2013

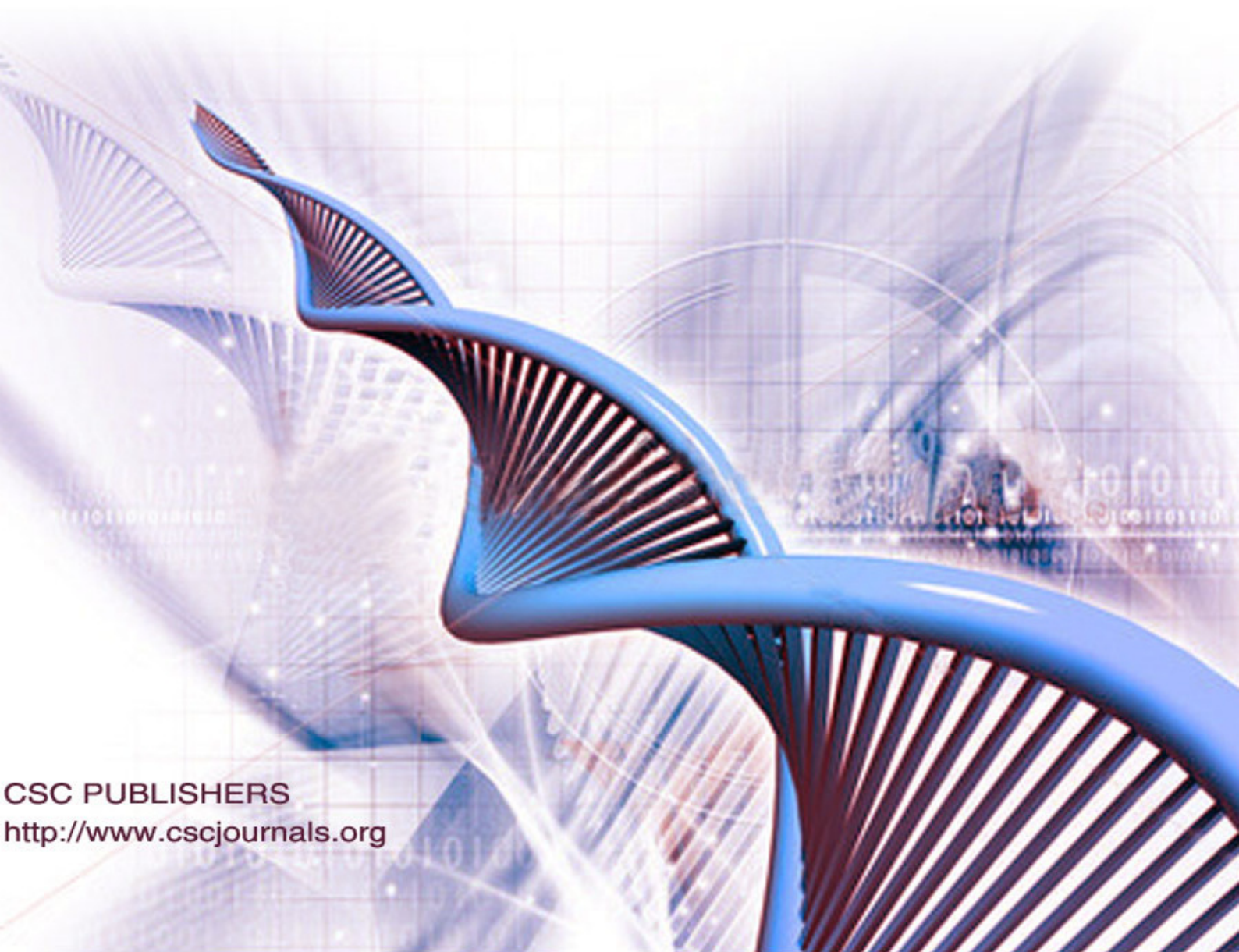
Editor-in-Chief
Professor João Manuel R. S. Tavares

INTERNATIONAL JOURNAL OF

BIOMETRICS AND BIOINFORMATICS (IJBB)

ISSN : 1985-2347

Publication Frequency: 6 Issues / Year



CSC PUBLISHERS
<http://www.cscjournals.org>

INTERNATIONAL JOURNAL OF BIOMETRICS AND BIOINFORMATICS (IJBB)

VOLUME 7, ISSUE 2, 2013

**EDITED BY
DR. NABEEL TAHIR**

ISSN (Online): 1985-2347

International Journal of Biometrics and Bioinformatics (IJBB) is published both in traditional paper form and in Internet. This journal is published at the website <http://www.cscjournals.org>, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJBB Journal is a part of CSC Publishers

Computer Science Journals

<http://www.cscjournals.org>

INTERNATIONAL JOURNAL OF BIOMETRICS AND BIOINFORMATICS (IJBB)

Book: Volume 7, Issue 2, September 2013

Publishing Date: 15-09-2013

ISSN (Online): 1985-2347

This work is subjected to copyright. All rights are reserved whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication of parts thereof is permitted only under the provision of the copyright law 1965, in its current version, and permission of use must always be obtained from CSC Publishers.

IJBB Journal is a part of CSC Publishers

<http://www.cscjournals.org>

© IJBB Journal

Published in Malaysia

Typesetting: Camera-ready by author, data conversion by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers, 2013

EDITORIAL PREFACE

This is the *Second* Issue of Volume *Seven* of International Journal of Biometric and Bioinformatics (IJBB). The Journal is published bi-monthly, with papers being peer reviewed to high international standards. The International Journal of Biometric and Bioinformatics is not limited to a specific aspect of Biology but it is devoted to the publication of high quality papers on all division of Bio in general. IJBB intends to disseminate knowledge in the various disciplines of the Biometric field from theoretical, practical and analytical research to physical implications and theoretical or quantitative discussion intended for academic and industrial progress. In order to position IJBB as one of the good journal on Bio-sciences, a group of highly valuable scholars are serving on the editorial board. The International Editorial Board ensures that significant developments in Biometrics from around the world are reflected in the Journal. Some important topics covers by journal are Bio-grid, biomedical image processing (fusion), Computational structural biology, Molecular sequence analysis, Genetic algorithms etc.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Started with Volume 7, 2013, IJBB appears with more focused issues related to biometrics and bioinformatics studies. Besides normal publications, IJBB intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

The coverage of the journal includes all new theoretical and experimental findings in the fields of Biometrics which enhance the knowledge of scientist, industrials, researchers and all those persons who are coupled with Bioscience field. IJBB objective is to publish articles that are not only technically proficient but also contains information and ideas of fresh interest for International readership. IJBB aims to handle submissions courteously and promptly. IJBB objectives are to promote and extend the use of all methods in the principal disciplines of Bioscience.

IJBB editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJBB. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJBB provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

Editorial Board Members

International Journal of Biometric and Bioinformatics (IJBB)

EDITORIAL BOARD

EDITOR-in-CHIEF (EiC)

Professor João Manuel R. S. Tavares
University of Porto (Portugal)

ASSOCIATE EDITORS (AEiCs)

Assistant Professor. Yongjie Jessica Zhang
Mellon University
United States of America

Professor. Jimmy Thomas Efirid
University of North Carolina
United States of America

Professor. H. Fai Poon
Sigma-Aldrich Inc
United States of America

Professor. Fadiel Ahmed
Tennessee State University
United States of America

Professor. Yu Xue
Huazhong University of Science and Technology
China

Associate Professor Chang-Tsun Li
University of Warwick
United Kingdom

Professor. Calvin Yu-Chian Chen
China Medical university
Taiwan

EDITORIAL BOARD MEMBERS (EBMs)

Assistant Professor. M. Emre Celebi
Louisiana State University
United States of America

Dr. Ganesan Pugalenth
Genome Institute of Singapore
Singapore

Dr. Vijayaraj Nagarajan
National Institutes of Health
United States of America

Dr. Wichian Sittiprapaporn
Mahasarakham University
Thailand

Dr. Paola Lecca
University of Trento
Italy

Associate Professor. Renato Natal Jorge
University of Porto
Portugal

Assistant Professor. Daniela Iacoviello
Sapienza University of Rome
Italy

Professor. Christos E. Constantinou
Stanford University School of Medicine
United States of America

Professor. Fiorella SGALLARI
University of Bologna
Italy

Professor. George Perry
University of Texas at San Antonio
United States of America

Assistant Professor. Giuseppe Placidi
Università dell'Aquila
Italy

Assistant Professor. Sae Hwang
University of Illinois
United States of America

Associate Professor Quan Wen
University of Electronic Science and Technology
China

Dr. Paula Moreira
University of Coimbra
Portugal

Dr. Riadh Hammami
Laval University
Canada

Dr Antonio Marco
University of Manchester
United Kingdom

Dr Peng Jiang
University of Iowa
United States of America

Dr Shunzhou Yu

General Motors Global R&D Center
United States of America

Dr Christopher Taylor

University of New Orleans
United States of America

Dr Horacio Pérez-Sánchez

University of Murcia
Spain

TABLE OF CONTENTS

Volume 7, Issue 2, September 2013

Pages

- 74 - 92 AR-based Method for ECG Classification and Patient Recognition
Branislav Vuksanovic, Mustafa Alhamdi
- 93 - 122 Face Images Database Indexing for Person Identification Problem
Jyotirmay Dewangan , Somnath Dey, Debasis Samanta

AR-based Method for ECG Classification and Patient Recognition

Branislav Vuksanovic

*Faculty of Technology/School of Engineering
University of Portsmouth
Portsmouth, PO1 2UP, United Kingdom*

branislav.vuksanovic@port.ac.uk

Mustafa Alhamdi

*Faculty of Technology/School of Engineering
University of Portsmouth
Portsmouth, PO1 2UP, United Kingdom*

mustafa.alhamdi@port.ac.uk

Abstract

The electrocardiogram (ECG) is the recording of heart activity obtained by measuring the signals from electrical contacts placed on the skin of the patient. By analyzing ECG, it is possible to detect the rate and consistency of heartbeats and identify possible irregularities in heart operation. This paper describes a set of techniques employed to pre-process the ECG signals and extract a set of features – autoregressive (AR) signal parameters used to characterise ECG signal. Extracted parameters are in this work used to accomplish two tasks. Firstly, AR features belonging to each ECG signal are classified in groups corresponding to three different heart conditions – normal, arrhythmia and ventricular arrhythmia. Obtained classification results indicate accurate, zero-error classification of patients according to their heart condition using the proposed method. Sets of extracted AR coefficients are then extended by adding an additional parameter – power of AR modelling error and a suitability of developed technique for individual patient identification is investigated. Individual feature sets for each group of detected QRS sections are classified in p clusters where p represents the number of patients in each group. Developed system has been tested using ECG signals available in MIT/BIH and Politecnico of Milano VCG/ECG database. Achieved recognition rates indicate that patient identification using ECG signals could be considered as a possible approach in some applications using the system developed in this work. Pre-processing stages, applied parameter extraction techniques and some intermediate and final classification results are described and presented in this paper.

Keywords: Electrocardiogram Classification, Individual Patient Recognition, AR Model, MIT/BIH Database.

1. INTRODUCTION

The electrocardiogram (ECG) represents a measure of the electrical activity of the heart. To measure this activity electrodes are placed on the skin in order to detect the bioelectric potentials given off by the heart that reach the skins surface. Studying the ECG signal can, in many cases, provide an insight into understanding life-threatening cardiac conditions [1]. These studies are usually concerned with detecting and classifying various types of arrhythmias, which can be defined as an irregular heartbeat rate or a disturbance in the regular heartbeat rate. Irregularities in the rhythm of the heart can indicate various causes including disease (e.g., coronary artery disease, diabetes, and cardiomyopathy), medications, an aging heart or metabolic problems.

Arrhythmia usually causes the heart to pump blood less effectively. While most of cardiac arrhythmias are temporary and benign, some arrhythmias may be life-threatening and require medical treatment. One of the most serious arrhythmias is sustained ventricular arrhythmia, usually caused by the damaged heart muscle [1]. This condition is dangerous because it may

degenerate into a totally disorganized electrical activity known as ventricular fibrillation, when the heart's action is so disorganized that it quivers and does not contract, thus failing to pump blood. It is therefore crucial for the patient to receive urgent medical attention when this type of arrhythmia occurs. Detection of ventricular arrhythmia can be done from the electrocardiogram (ECG) signal. It should however be mentioned that a normal ECG does not rule out serious heart disease and that some heart attacks cannot be detected by ECG. The same potential difference on the body surface for example can be due to different configurations of sources; hence, abnormal activity may be masked. Continuous observation and detection of abnormal ECG signals can be difficult due to a large number of patients in intensive care units. In addition to a simple ECG test, a longer recording using a portable Holter monitor worn by the subject during a 24 to 48 hour period can be made [2]. The Holter monitor results are passed to a cardiologist who needs to examine the recordings and determine a diagnosis. Examination of these recordings is a time-consuming process and an automated processing of the ECG that assists the cardiologist in determining a diagnosis would be of assistance. A number of different methods for automated arrhythmia detection have been developed in the past few decades in attempt to assist with the ECG monitoring task. Most of the methods report recognition of heart condition with a certain level of accuracy. This paper proposes an improved method that can achieve high classification accuracy on the test signals available from the web-based ECG database (MIT/BIH) and Milano VCG/ECG database. Most common arrhythmia detection methods rely on QRS detection [3][4] and beat classification using a number of classification techniques. A QRS complex, including other most important features of the ECG signal, is shown in FIGURE 1 using an idealised plot of a single heart beat with indicated main intervals and segments during the heart activity.

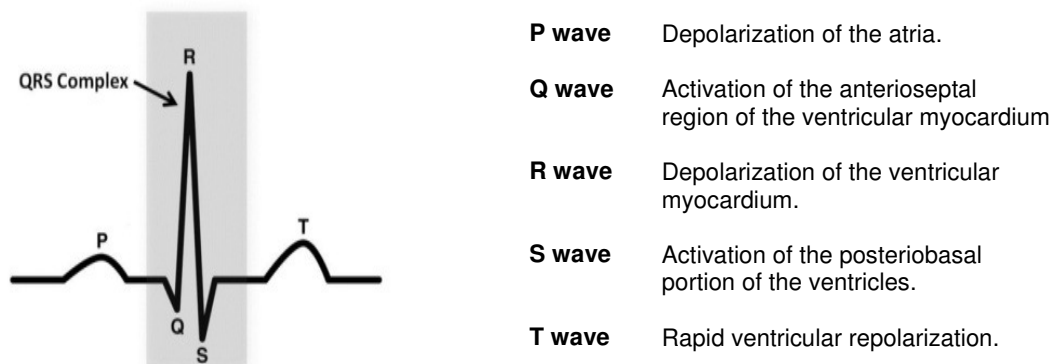


FIGURE 1: Ideal ECG Signal with Indicated Key Features.

QRS or beat detection is the crucial part of almost any ECG processing algorithm. QRS is a major feature of ECG signal, caused by ventricular depolarization of the human heart. Once the positions of the QRS complexes are found, the locations of other components of ECG can be found relative to those positions and cardiac period established. In many cases simple pre-processing and thresholding of the heart rate can be sufficient for correctly identifying many arrhythmias. However, the QRS pattern recognition techniques can also be severely affected by noise due to unfavourable signal acquisition conditions so other, more advanced, approaches have been developed.

The ECG features can be extracted in time domain [4] or in frequency domain [5] using more advanced feature extraction methods. Wigner-Ville analysis in a two-dimensional frequency domain has also been proposed to address the lack of spectral features and non-stationary behaviour of ECG signals [6]. Some other recently implemented methods include Discrete Wavelet Transform [7][8], Karhunen-Loeve Transform [9], Hermitian Basis [10] and other techniques [11]. Paper [8] uses wavelet transform to decompose the ECG signal into elementary

building blocks well localised in time to detect QRS complex. Detection accuracy reported in [8] is 99.8% despite noise, baseline drift and other artefacts present in majority of ECG signals. In [11] the QRS beats were obtained as 29 point templates. 14 points on either side of the main peak were used to form this template in the first stage of the process but those dimensions were then reduced using principal-component analysis (PCA) also known as Karhunen-Loeve Transform. Reduced number of “effective” features was obtained by discarding the linear combinations with small variance and retaining terms with large variance to represent a template used for QRS complex detection. In [12] a method to detect QRS complex using a delineation function defined via an envelope of the ECG signal was reported. This method yields a single positive pulse for each complex and uses a delineation function to define the onset and end of the QRS with a high accuracy. Hermitian Basis representation of the QRS complexes was later proposed in [10] resulting in a set of parameters that can be used to accurately represent the QRS complex detected in this way. Parameters extracted using this approach can be used in various other applications including data compression. The Hermite Basis approach also provides a width parameter to describe the QRS complex and can therefore describe beats with large differences in QRS duration while the Karhunen-Loeve transform approach encounters problems in those cases. This approach is further developed in [13] to include the multiple-input adaptive linear combiner, using as inputs the succession of the QRS complexes to estimate parameters of Hermitian Basis including the estimation of the width related parameter.

Various machine learning algorithms can then be applied to classify the ECG signals according to the features and information extracted. Machine learning algorithms used for ECG classification include Bayesian [14] and heuristic approaches [15], expert systems [9], Markov models [16], self-organizing map [10], and Artificial Neural Networks [17][18][19][20]. Naïve Bayes is one of the simplest probabilistic classifiers. The model constructed by this algorithm is a set of probabilities where each member of this set corresponds to the probability that a specific feature f_i appears in the instances of class c , i.e., $P(f_i|c)$. The Naive Bayes classifier is known to be optimal when all features describing the class c are independent of each other although several studies have shown Naive Bayes to be competitive with more sophisticated classifiers [21] even when the clear dependence amongst the variables in the class does exist. Artificial neural networks are mathematical models for information processing based on the biological neural complexes. Both Back Propagation (BP) and Radial Basic Function (RBF) networks are well-known variants of neural networks and have been used in various tasks of classification of biomedical signals. Performance of BP and RBF networks in classification tasks using ECG and blood pressure data has been investigated [22]. RBF network using K-nn means clustering algorithm as a basis function was proven to result in slightly higher classification accuracy for cardiac diagnosis compared to BP networks. In [23] AR modeling was performed on ECG data from normal ECGs as well as various arrhythmias. The AR coefficients were computed using Burg's algorithm and classified using a generalized linear model (GLM) based algorithm. AR modeling results showed that an order of four was sufficient to accurately model majority of ECG signals. The accuracy of detecting and classifying heart conditions was 93.2% using the GLM based classification algorithm. The research reported in [24] demonstrates that the all pole, low order AR model can be used to construct a feature space for accurate classification of underwater passive sources, combining it with the simple K-nn classifier. ECG features can also be extracted using this approach and combined with different classification algorithms to achieve high classification rate. Quadratic Discriminant Function (QDF) based algorithm has recently been used to classify certain cardiac arrhythmias with 97% classification accuracy [25] indicating significant improvement compared to most of the previously reported results.

In general, the ECG classification results are strongly determined by two main factors – derived set of heartbeat features and selection of techniques employed to recognise and classify those features. The approach and techniques adopted in this work are described and discussed in the rest of the paper. Section 2 lists the main stages in the system and discusses techniques and function of each system block, while the Section 3 presents the intermediate and final classification results achieved with the system. Results and some further work and ideas are summarized and outlined in the concluding section.

2. METHOD

The method developed in this work consisting of four major stages proposed to classify the patients according to recorded ECG signals. Those stages are: pre-processing of the raw ECG signals to reduce noise and various other artefacts present in the signal, QRS detection, AR parameter extraction and classification of extracted parameters and corresponding signals.

2.1 Pre-processing

Real ECG signals are usually non-stationary, containing slow linear drifts or more complex trends. Causes of those trends are explained in details in [26] but the two most important factors can be considered to be respiratory modulation and the baseline drift. The baseline drift of the ECG signal is mostly generated due to the variation of interaction between the sensor and the body. To enable further analysis of the ECG signals various methods to remove those trends have been used in the past. Those have mainly concentrated on removing slow, nonstationary trends from the ECG signals. Various methods to remove those trends have been developed [26]. In this work a recently reported method based on smoothness priors approach [27] has been used. Using this approach the trend component z_{trend} of the ECG signal is modelled using linear observation model. The RR interval series of ECG signal can be represented as:

$$z = (R_2 - R_1, R_3 - R_2, \dots, R_N - R_{N-1})^T \in \mathbf{R}^{N-1} \quad (1)$$

where N is the number of R peaks detected. The RR series can be considered to consist of two components:

$$z = z_{\text{stat}} + z_{\text{trend}} \quad (2)$$

where z_{stat} is the nearly stationary RR series of interest and z_{trend} is the low frequency aperiodic trend component. The trend component can be modeled with a linear observation:

$$z_{\text{trend}} = H\theta + v \quad (3)$$

Where $H \in \mathbf{R}^{(N-1) \times M}$ represents the observation matrix, $\theta \in \mathbf{R}^M$ contains the regression parameters and v is the observation error. To obtain the estimate of the regression parameters $\hat{\theta}$, the regularised least square approach is used:

$$\hat{\theta}_\lambda = \arg \min \{ \|H\theta - z\|^2 + \lambda^2 \|D_d(H\theta)\|^2 \} \quad (4)$$

resulting in:

$$\hat{\theta}_\lambda = (H^T H + \lambda^2 H^T H_d^T D_d H)^{-1} H^T z \quad (5)$$

where λ is the regularization parameter and D_d indicates the discrete approximation of the d 'th derivative operator. It has been shown [27] that this method operates as a time-varying FIR high-pass filter where the cut-off frequency of the filter decreases when λ increases.

The detrended ECG signal, z_{trend} can now be obtained as:

$$z_{\text{trend}} = z - z_{\text{stat}} = H\hat{\theta}_\lambda = z(I - (I + \lambda^2 D_2^T D_2)^{-1}) \quad (6)$$

This signal is further filtered through the band-pass filter before the detection of QRS complex is attempted.

2.2 QRS Detection

The single most important feature of ECG signal is the QRS complex. As indicated in FIGURE 1. all other features, the P and T waves as well as the onset and offset of the QRS complex are defined relative to the QRS complex. The P and the T wave occur before and after the QRS

complex respectively. Without the accurate knowledge of the QRS location, P and T waves are hard to detect and distinguish from each other. Most of the QRS detection methods depend heavily on filtering stage followed by averaging according to a threshold value. This threshold value is used to distinguish between noise signal and the QRS complex, It can be chosen based on the peak height or peak location of the ECG signal [28]. There are other methods depending on the machine learning algorithms [28] like the P-spectrum method [29] which is a robust method for periodicity detection based on the data singularity.

In order to accurately detect the positions of QRS complex in the recorded ECG signals, filter bank analysis method [30] was used. This approach employs a bank of linear phase filters to decompose the ECG signal into subbands with uniform frequency bandwidths in order to account for the ECG signal energy distribution in the frequency domain. A number of features related to QRS complex are extracted from individual subbands and combined to indicate the position of the QRS complex in the analysed signal. The filterbank used to analyse ECG signals contains M analysis and M synthesis filters, each of length L and is used to produce the subband signals $U_l(z)$ by filtering the input signal $X(z)$:

$$U_l(z) = H_l(z)X(z) \quad l = 0, 1, \dots, M - 1 \quad (7)$$

where $H_l(z)$ represents the transfer function of each bandpass filter. After down sampling, each (down sampled) signal can be expressed as:

$$W_l(z) = \frac{1}{M} \sum_{k=0}^{M-1} U_l\left(\frac{1}{M} W^k\right) = \frac{1}{M} \sum_{k=0}^{M-1} H_l\left(\frac{1}{M} W^k\right) X_l\left(\frac{1}{M} W^k\right) \quad (8)$$

where $l = 0, 1, \dots, M - 1$, $W = e^{-j\left(\frac{2\pi}{M}\right)}$. A variety of features, indicative of the QRS complex have been extracted from the signal by combining the subbands of interest. For example, feature P_1 corresponding to the energy in those subbands 1, 2 and 3, can be computed as:

$$P_1 = \sum_{l=1}^3 |W_l(z)| \quad (9)$$

Similarly, features P_2 and P_3 , computed for the subbands $\{1, 2, 3, 4\}$ and $\{2, 3, 4\}$, are proportional to the energies in the corresponding bands. A sum-of-squares feature P_4 is computed using the following equation:

$$P_4 = \sum_{l=1}^3 (W_l(z))^2 \quad (10)$$

Heuristic beat detection logic uses these features to identify positions of QRS complexes in the ECG signal. Detection system consists of a number of sequential levels of logic designed to maximise the number of true positives (TPs). For this purpose multiple detectors are operated simultaneously and the results of each detector are fused together to arrive at the final decision about the beat positions in the signal.

First level serves as an “event detector” and uses a moving window integrator (MWI) which averages two samples of a particular feature (P_1 for example) at the downsample rate. Inflection points in the output of this MWI are then used to identify possible beat candidates for the beats as peaks in the MWI output. These candidates then go through the further logic in level 2 designed to eliminate large number of false positives (FPs), events inaccurately identified as beats by level 1, mostly introduced by the presence of noise in the signal. This level operates two one-channel beat detection blocks which have complementary FN and FP detection rates with outputs finally

combined in level 3 of the detector by incorporating a set of if-then-else rules. If channels 2 in level 2 indicates the beat than the output of level 3 classifies the current event as a beat. If channel 1 indicates a beat and channel 2 indicates not-a-beat detection strengths of each channel are compared and the final decision is made based on this comparison. Level 4 is used as one more check before the final decision about the presence of the beat in the ECG signal is made. This level uses P_3 as the input to MWI to confirm decision made at level 3 and reduce the rate of FNs introduced at previous levels. After beat occurs there is a physiological refractory period of about 200 ms before another can occur. Level 5 uses timing information of the ECG signal to eliminate possible FPs during the refractory period which further improves the accuracy of the beat detection in this system through the partial blanking of the refractory period.

Approach to detect individual beats in the recorded ECG signal is implemented in Matlab function “nqrsdetect” published and available on the web. Function can be applied using syntax: “QRSs=nqrsdetect(EGCsignal, Fs);” where F_s represents the sampling frequency of the analysed ECG signal (“EGCsignal”). Upon the execution of this function, vector “QRSs” contains the positions of detected R peaks in the signal.

2.3 Parameter Extraction

Feature selection and extraction is one of the crucial stages in the classification system.

In this work a simple approach of modelling two or more successive ECG beats, using a discrete form of an autoregressive (AR) signal model of order p , $AR(p)$, has been applied. ECG beats are detected using filterbank method briefly explained in the previous section. Coefficients of the estimated AR model are then used as features suitable for signal classification in the final stage of the system.

Using an AR model, a signal sequence $y(t)$, extracted group of ECG beats in this case, can be represented by the relationship:

$$y(n) = a_1y(n-1) + a_2y(n-2) + \dots + a_p y(n-p) + \varepsilon(n) \quad (11)$$

where $a_k = (1, 2, \dots, p)$ are the model coefficients, also known as autoregressive parameters, used in the classification process and the $\varepsilon(n)$ is a white noise series, innovation process with zero mean and variance σ^2 . An estimated autoregressive model of the same order p can then be written as:

$$y(n) = \hat{a}_1 y(n-1) + \hat{a}_2 y(n-2) + \dots + \hat{a}_p y(n-p) + \hat{\varepsilon}(n) \quad (12)$$

where $\hat{a}_k = (k = 1, 2, \dots, p)$ are the estimated parameters of the autoregressive model and $\hat{\varepsilon}(n)$ are the estimated innovations. The estimated autoregressive model can be interpreted as the p -point prediction filter where value of the output $y(n)$ is estimated from the previous $p-1$ output values of the AR process:

$$\hat{y}(n) = \sum_{i=1}^p \hat{a}_i y(n-i) \quad (13)$$

As samples $y(n)$ can not be predicted exactly a modelling error is introduced. This error or residue corresponds to difference between the measured and the estimated values and is in fact equal to the value of the estimated innovation:

$$r = y(n) - \hat{y}(n) = \hat{\varepsilon}(n) \quad (14)$$

A number of methods can be used to estimate autoregressive parameters of the AR model. Most often used are the least-squares approach (LS), the Yule-Walker approach (YW) and Burg's

method [31]. Least-squares method minimises the total squared residue over data samples $p + 1$ to N , which leads to a system of linear equations:

$$\begin{bmatrix} C_{11} & C_{12} & \dots & C_{1p} \\ C_{21} & C_{22} & \dots & C_{2p} \\ \dots & \dots & \dots & \dots \\ C_{p1} & C_{p2} & \dots & C_{pp} \end{bmatrix} = \begin{bmatrix} \hat{a}_1 \\ \hat{a}_1 \\ \dots \\ \hat{a}_1 \end{bmatrix} = - \begin{bmatrix} C_{01} \\ C_{02} \\ \dots \\ C_{0p} \end{bmatrix} \quad (15)$$

C_{ij} elements of the matrix in the above equation represents an unbiased estimate of the autocovariance function for delay $i - j$:

$$C_{ij} = \frac{1}{N - p} \sum_{n=p+1}^N y(n - i)y(n - j) \quad (16)$$

Yule-Walker method includes the first and last p data points which results in matrix equation:

$$\begin{bmatrix} \hat{R}_0 & \hat{R}_1 & \dots & \hat{R}_{p-1} \\ \hat{R}_1 & \hat{R}_0 & \dots & \hat{R}_{p-2} \\ \dots & \dots & \dots & \dots \\ \hat{R}_{p-1} & \hat{R}_{p-2} & \dots & \hat{R}_p \end{bmatrix} = \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \dots \\ \hat{a}_p \end{bmatrix} = - \begin{bmatrix} \hat{R}_1 \\ \hat{R}_2 \\ \dots \\ \hat{R}_p \end{bmatrix} \quad (17)$$

where elements of the matrix in the (18) equation represent the biased estimate of the autocovariance function:

$$\hat{R}_k = \frac{1}{N} \sum_{n=k+1}^N y(n)y(n - k) \quad (18)$$

While both LS and YW methods estimate the autoregressive parameters directly, using the autocorrelation matrix of the signal sequence, Burg’s method first finds the reflection coefficients of the equivalent lattice structure predictor filter based on the least squares criteria. From these, the AR parameter estimates are determined using the Levinson-Durbin algorithm [32]. The reflection coefficients constitute unbiased estimates of the partial correlation coefficients. Each of the described algorithms above has its own drawbacks and advantages and is used for various applications such as spectral analysis. In most cases, the algorithms result in similar estimated values in most situations. But it has been suggested [33] that Burg’s algorithm might be preferable due to poor estimates by YW in some cases and the possible instability of the least squares model.

In the previous discussion, it is assumed that the model order p is known which is not the case in practice. In fact, determination of the model order can be one of the most difficult tasks in time series modeling. In a number of situations, prediction error power for various model orders can be obtained and used as an indicator for the sufficient model order. Another option is to use the Aikike information criterion [34] to determine the order of the AR model. In this work, model order was determined by observing the power of modeling error for various model orders. Orders of $p = (2,3,4,5)$ have been found to yield satisfactory results for the pre-processed ECG signals. Further in the paper a set of results obtained for $p=2$ and 3 model order is presented. It is also worth pointing that more complex models have been used for time series modeling, prediction and even classification tasks in the past. Autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) models [35] are composed of different terms in addition to autoregressive (AR) term. While AR term includes the influence of lagged observed values as already explained, ARMA model combines it with the moving average (MA) term which describes the forecasting errors using the following form:

$$Y(n) = a_1y(n-1) + a_2y(n-2) + \dots + a_p y(n-p) + b_0\varepsilon(n) + b_1\varepsilon(n-1) + \dots + b_r\varepsilon(n-r) \quad (19)$$

where b_k ($k = 1, 2, \dots, r$) are the MA model coefficients. Autoregressive (AR) part of this model performs well when the signals with the narrowband spectra need to be modeled while the MA model provides a good approximation for those spectra which are characterized by broad peaks and sharp nulls. However, the problem of estimating parameters of MA model is basically a non-linear one, and is significantly more difficult to solve than the AR parameter estimation problem. Since ARMA model combines both AR and MA models, difficulties in MA and ARMA estimation problems are similar and could be avoided if a simpler AR can approximate ECG signal with satisfactory accuracy.

In addition to ARMA model, the ARIMA model also includes an integrating term (I) which can account for the non-stationary of the series. ARIMA model first removes the trends and various cyclic features from the signal that are beyond the capacity of stationary ARMA model which can then be used to model the remaining detrended and depersonalized signal. Use of ARIMA model for ECG beats modeling is currently considered in the continuation of this work, in order to improve the accuracy of patient recognition from short sections of obtained ECG signals. Burg algorithm to estimate the parameters of an AR model is implemented and available in the Matlab System Identification toolbox via “ar” function, usually used in combination with the “iddata”. This function creates an “iddata” type object from the given time series in order to analyze it with the “ar” function. A short sequence of the Matlab program given in TABLE demonstrates a method used to estimate AR parameters from the part of ECG signal using “ar” function.

```

beat = ECGsignal(QRSs(p):QRSs(p+nb-1));    % extract nb beats from the ECG signal section
data = iddata(beat);                       % convert to "iddata" type object
m = ar(data,mo);                           % calculate AR parameters
ARs = polydata(m);

```

TABLE 1: Estimation of AR Parameters using “ar” Function in Matlab.

2.4 Classification

Various classification algorithms can be used to classify the extracted ECG signal features. In this work, features are represented by multidimensional vectors containing autoregressive coefficients calculated individually for each beat or pair of beats of the measured ECG signal. Effectiveness of classification methods depends on how well the vectors of features can be separated in the feature space. In many cases, where dimensionality of feature vectors is high, various algorithms can be used to reduce the size of the feature vectors. In this work, only a small number of autoregressive coefficients were found to represent a satisfactory model for most ECG signals in the database so no dimensionality reduction scheme was used.

The classifying methods proposed during the last decades include, Fuzzy Logic methods [36], Artificial Neural Network, Hidden Markov Model [16], Genetic Algorithm [37], Support Vector Machines, Self-Organizing Map, Bayesian [38] and other with each approach exhibiting its own advantages and disadvantages [39]. Algorithms used for ECG classification can mainly be categorized as either heuristic or statistical classification methods [15]. While heuristic approach tries to emulate the reasoning of the qualified cardiologist and the cardiologist provides the knowledge to construct a classifier, for statistical approach, probability densities of diagnostic features are estimated from a learning set of ECG features and a various multivariate techniques are then used to achieve classification. Main criterion for selection of particular classification method is the classification performance but other aspects should also be considered [15]. Statistical classifiers are considered in this work as they usually require less involvement of skilled operator or cardiologist. Main objective of the statistical approach is the allocation of an ECG to one group of diagnostic categories with minimum probability of misclassification.

Statistical methods used in this work are k-nearest neighbor (k-nn) classifier and linear and quadratic discriminant analysis based classifier [40][41]. Both methods belong to a group of so-called supervised learning methods, where some knowledge about data is available and used to produce an inferred function, classifier.

Because of its simplicity k-nn rule is one of the most often used methods in bioinformatics and other areas but care must be taken in selecting the model order as well as different distance metrics. Another important issue related to the use of k-nn is a complexity issue which can be relatively high if a training set of vectors is large. Linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) methods have been used in a large number of bioinformatics projects. If the data to be classified is not linearly separable, it is then advisable to use QDA method but it should be noticed that the capability of QDA to handle nonlinear data is still limited since it only considers the positive correlation between the variables. If the classification between two classes depends on the negative correlations between the data than noise rather than true information is introduced by QDA in the classification process.

Each of the mentioned methods is implemented in the Matlab Statistics Toolbox via "classify" function. LDA and QDA algorithms can be applied by setting the type option to "linear" or "quadratic" when this function is used. k-nn method is implemented via "knnclassify" function.

3. PROCEDURE AND RESULTS

To evaluate the performance of the proposed techniques, ECG data set containing three different types of ECG signals was used. Data set included normal ECG signals (NR) from the Politecnico of Milano VCG/ECG Database on Young Normal Subject [Politecnico Biosignals Archives [42], arrhythmia (AR) from the MIT-BIH Arrhythmia Database and ventricular arrhythmia (VAR) from the MIT-BIH Malignant Ventricular Arrhythmia Database [43]. Each type was represented by 20 half-hour excerpts of two-channel ambulatory ECG recordings, but 10 minutes per patient have been considered in this work. The time series related to the normal subjects were acquired with a sampling frequency $F_s = 500\text{Hz}$, while the time series for arrhythmic patients have $F_s = 250\text{Hz}$. $F_s = 360\text{Hz}$ is the sampling frequency for Ventricular Arrhythmia patients (both Ventricular tachycardia and Ventricular fibrillation).

The whole system used for the first project task of ECG arrhythmia detection and classification is summarized in the block diagram given in FIGURE 2. The raw ECG signals are first downsampled to the lowest sampling frequency of the ECG signals from the set (250 Hz in this case). The signals are then processed by smoothness priors method to detrend them and to improve the accuracy of subsequent algorithms implemented in the system. The detrended signal is then passed through the band-pass filter to remove the other types of noise that can still be present in the signal. Suitable cut-off frequencies of this filter have been determined after some experimentation. For the results presented in this section, the 6-th order Butterworth filter with the lower and upper cut-off frequencies set to 5 and 40 Hz respectively was used. The peak detection algorithm based on the filterbanks is implemented in the next stage. Using detected peak positions, each ECG signal is then split in the groups of 1, 2 or more successive beats and AR parameters are extracted for each obtained group. Finally, classification of extracted features using described classification algorithms is performed.

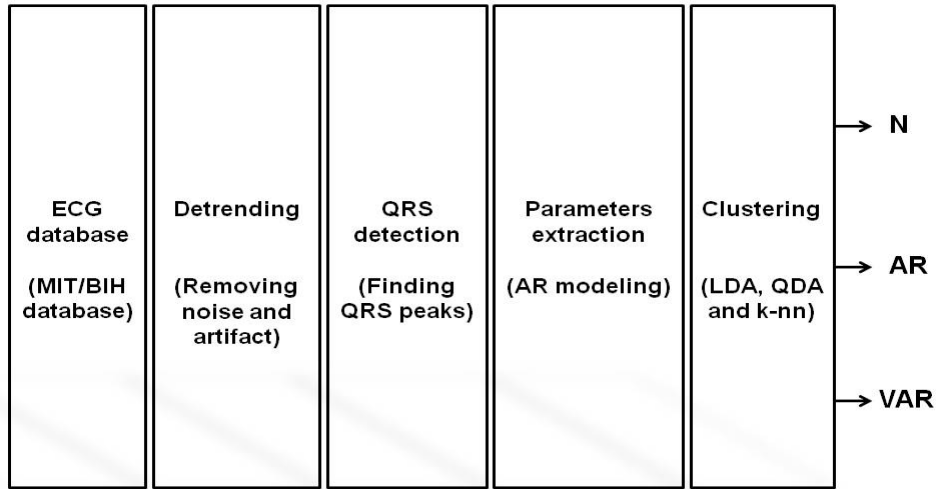
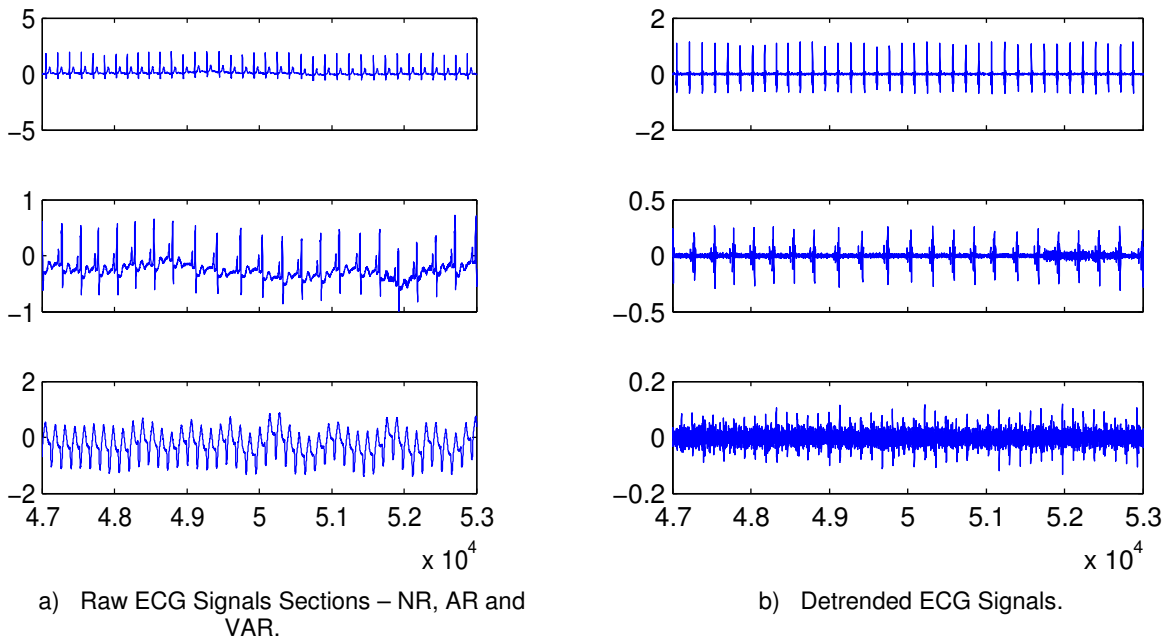
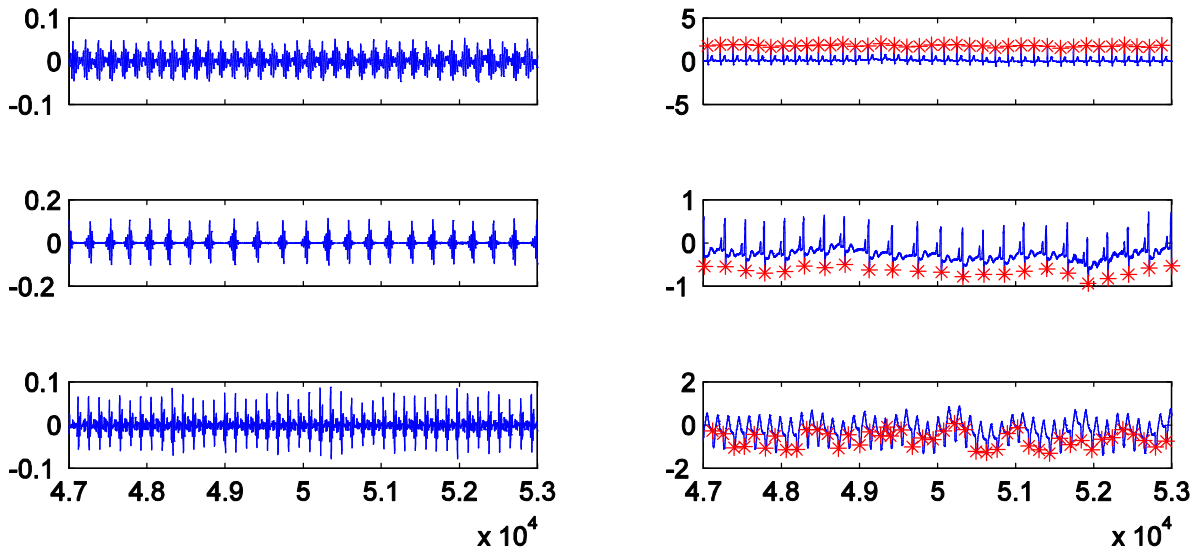


FIGURE 2: Stages in the AR based ECG Classification System.

Number of beats in the group as well as the number of AR parameters extracted for each group has a significant effect on the performance of the classification stage of the system, so some experimenting with those parameters has been performed. Good classification rate has been achieved for 1-5 beats in the group and 2-4 AR parameters used to model each section of ECG signal. In this paper, we show results for 2 beats per group and $p=2$ and 3 AR orders. ECG signals at various pre-processing stages in the system are shown in FIGURE 3. Detected QRS complexes for the section of one ECG signal from each group are indicated in FIGURE 3c.

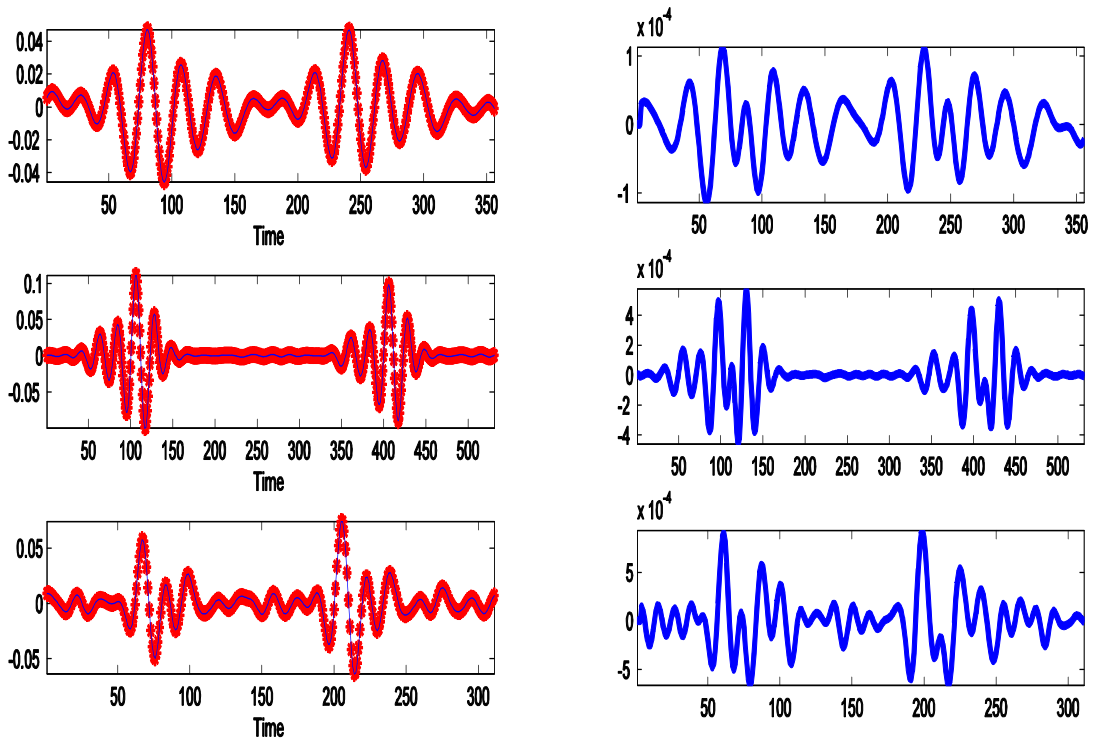




c) BP Filtered and Detrended ECG Signals.

d) Raw ECG Signals with Detected QRS Complexes.

FIGURE 3: Sample ECG signals at various stages in the AR based ECG classification system.



a) Model and Original (pre-processed) ECG Signal Section.

b) Modeling Error.

FIGURE 4: Results of a third order AR modeling on signal section consisting of 2 beats for each ECG type.

Individual section containing two beats, extracted from pre-processed ECG signal and the corresponding AR(3) model are shown in FIGURE 4. Obtained signal model has been plotted over the original signal section previously detrended and filtered as indicated in FIGURE 2. Accuracy of obtained AR models is high, as indicated by the modeling errors, residuals shown in FIGURE 4 b) for the corresponding sections and obtained models. Following beat detection and signal segmentation stage, AR parameters are extracted for each extracted group of beats. Numbers of beats in the group as well as the number of AR parameters extracted for each group have some influence on the classification algorithm performance. Good classification rate has been achieved for 1-5 beats in the group and 2-4 AR parameters used. In this paper, we show results for 2 beats per group and AR order p of 2 and 3. FIGURE 5 shows the modeling error for different orders of AR model. This figure can generally be used to determine the optimal order of the AR model, which is usually selected at the break point (“knee”) of the plot. For comparison purposes, two plots for ECG signals with and without pre-processing are shown. The breakpoint in the plot can easily be determined for processed ECG signal (2 or 3) while it is more difficult to pinpoint the equivalent position in the plot given for raw ECG. It is also worth noting significantly lower modeling error for the processed signal compared to the modeling error for the same but unprocessed signal section.

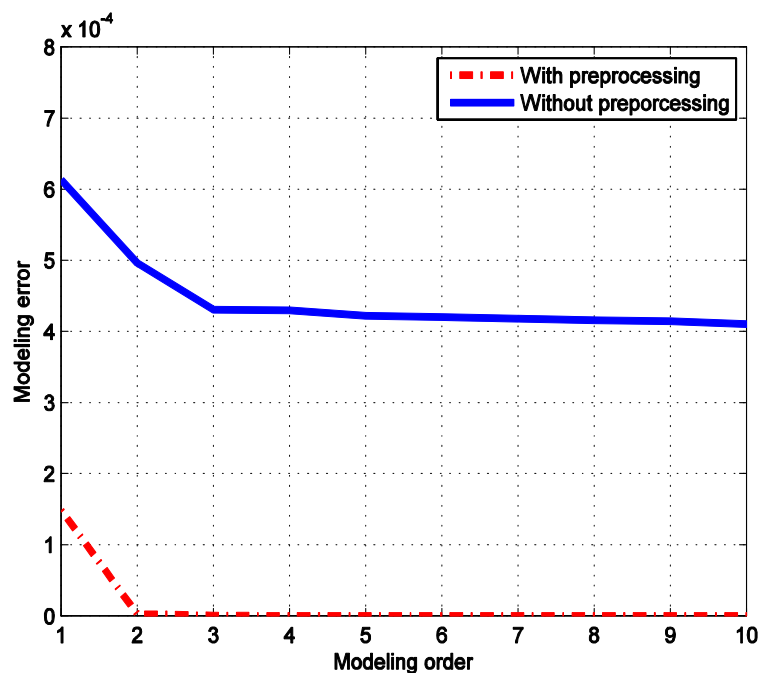
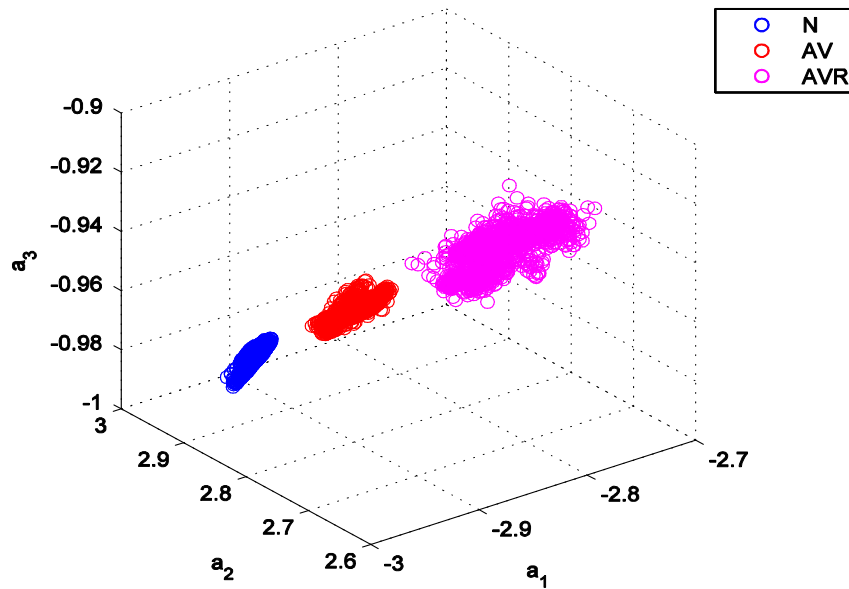


FIGURE 5: Modeling error for different AR model orders with “knee point” usually used for model selection.

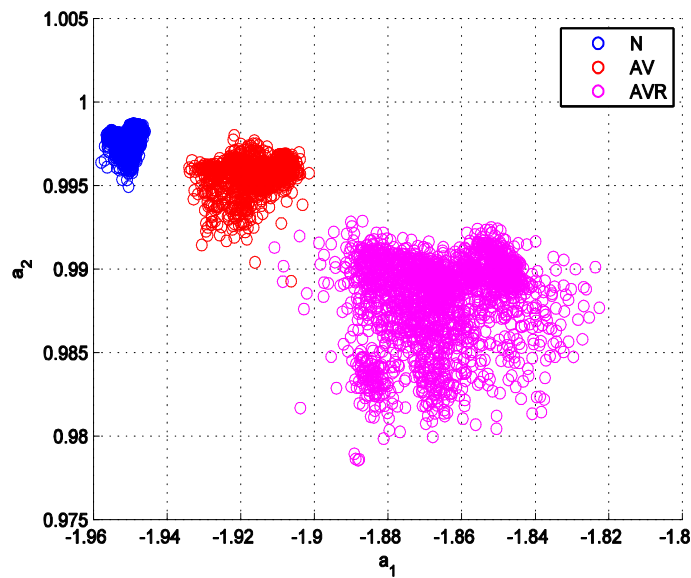
Extracted AR parameters, for each group of 2 QRSs for all signals in the data set are plotted in FIGURE 6 where good separation between 3 data clouds can be observed for both 3D, where AR order = 3 and 2D scatter plots, where AR order = 2, thus enabling accurate classification of each extracted parameter set. A number of described classification algorithms were tested on this set (k-nn, LDA, QDA, Bayes) each resulting in a completely correct, error-free classification of each parameter set. It is also interesting to note tightness of the cluster corresponding to normal ECG signals in contrast to progressively more scattered clouds of parameters related to arrhythmia and ventricular arrhythmia conditions.

Developed algorithm has also been tested on the extension of ECG database, using ECG signals obtained from another 20 patients in each of three classes including normal, atrium fibrillation and

arrhythmia type ECG signals. FIGURE 7 shows the 3D- feature space formed by order 3 AR coefficients extracted from this ECG database.



a) Classification Results for Order 3 AR Coefficients.



b) Classification Results for Order 2 AR Coefficients.

FIGURE 6: Feature space of extracted AR coefficients indicating good separation between three groups of ECG signals a) 3D feature space, b) 2D feature space.

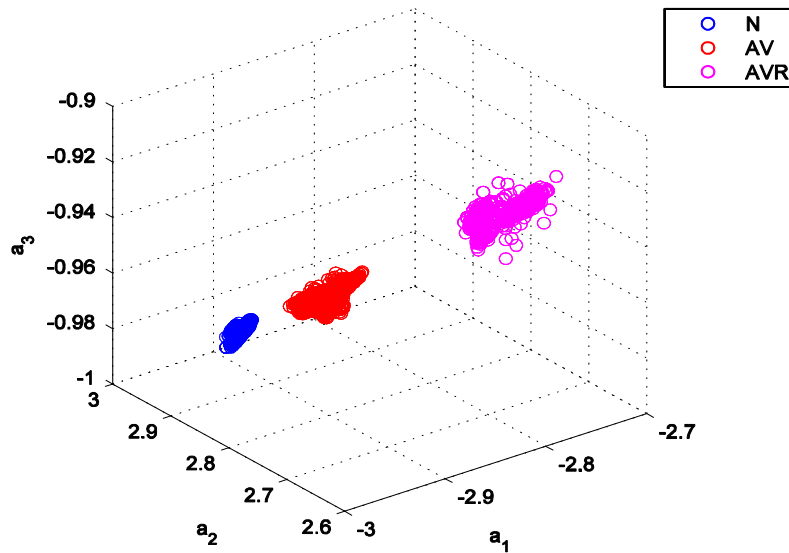


FIGURE 7: 3D-feature space of extracted AR coefficients resulting in completely accurate classification.

To indicate the importance and suitability of the pre-processing methods applied in this work, the rest of this section presents the equivalent set of results, 3D and 2D feature space plots obtained from the raw ECG signals (i.e. no filtering and detrending but including QRS detection). FIGURE 8 shows the results of AR modelling on automatically extracted groups of beats of raw ECG. It can be seen that although relatively good model is achieved it is still not as accurate compared to modelling results achieved with pre-processed ECG signal. More importantly, parameters of the model are poorly separated in 3D and 2D feature space giving no foundation for successful clustering and subsequent recognition of various conditions with any of tested classification methods. FIGURE 9 for example shows the results for $p=2$ and 3 model order.

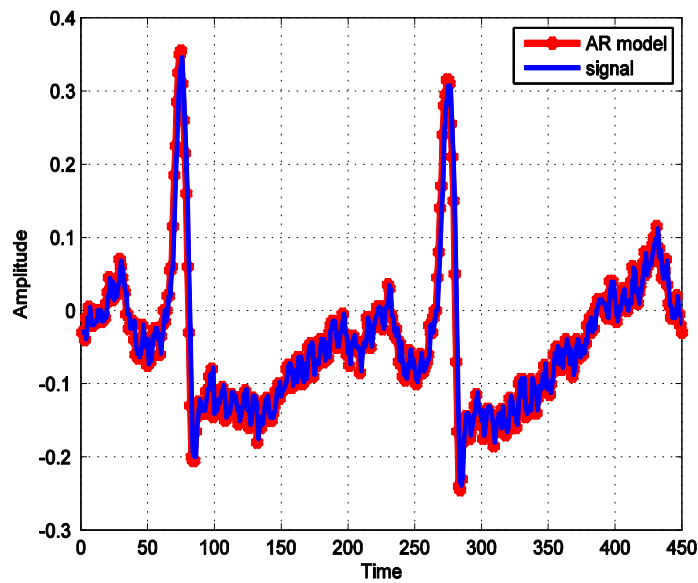
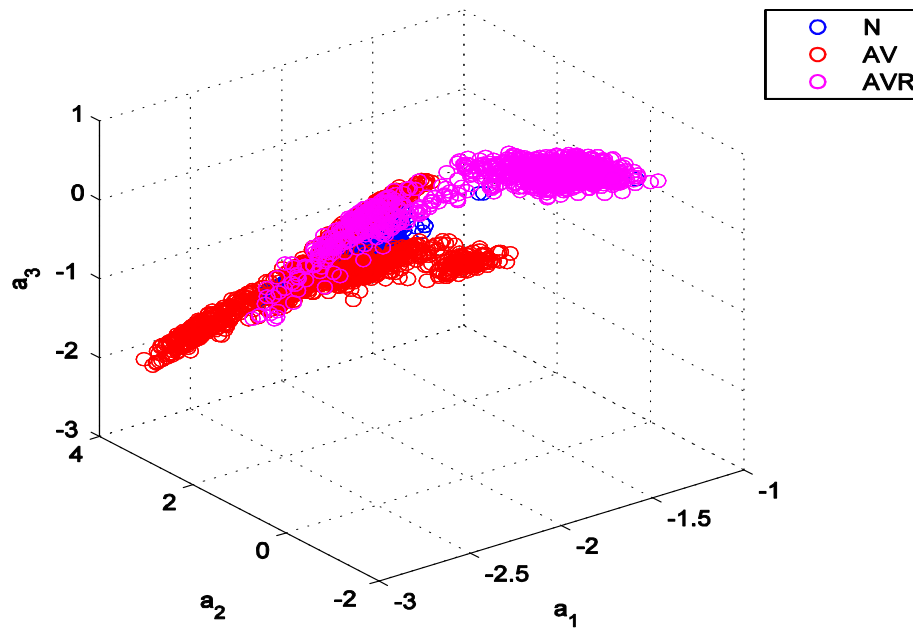
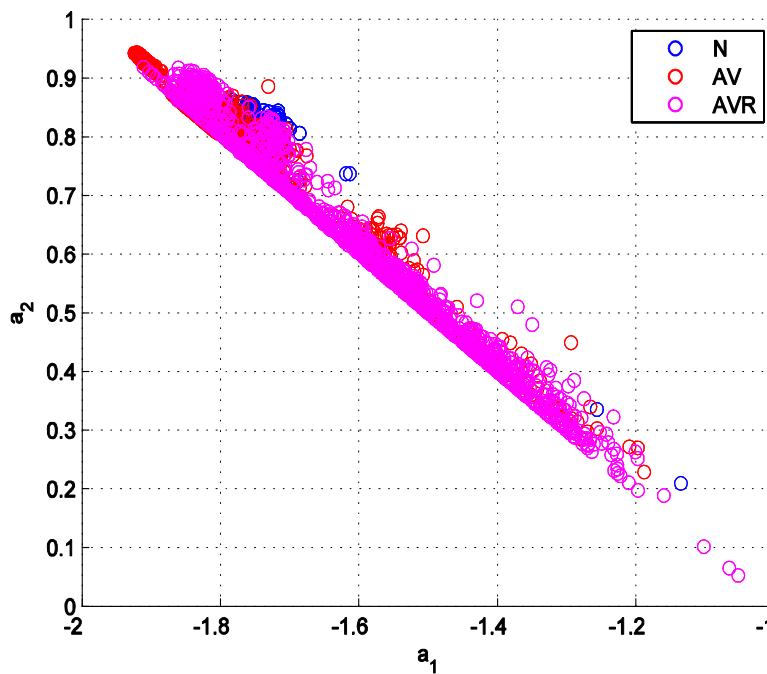


FIGURE 8: Results of AR modeling on automatically extracted groups of beats for raw ECG signal (i.e. no preprocessing stage).



a) Classification Results for Order 3 AR Coefficients.



b) Classification Results for Order 2 AR Coefficients.

FIGURE 9: 3D and 2D-feature space of extracted AR coefficients without any pre-processing steps.

In the reminder of this section, another interesting application of this method – patient identification from recorded ECG signal, is discussed and results summarized in

TABLE 2. Here, each individual section extracted from each ECG signal from the set has been classified into one of q different groups where q corresponds to the number of patients used in the experiment ($q = 20$ for each condition for the experimental set used in this work). 10-fold cross-validation technique was used to estimate performance of LDA and QDA classification techniques in this task. First part of the table gives the result achieved with the original feature set, i.e. the set containing AR parameters only. Second part of the table contains recognition results achieved with extended feature set, obtained when the power of modeling error, i.e. residual signal is used as an additional classification feature. Improvement in recognition rate is notable and indicates that the modified method using additional dimension in the feature set can be considered in patient identification tasks. It is also worth noting the improvement and higher rate of recognition achieved in the normal (N) and arrhythmia (AR) groups which indicates the possibility of using this approach to complement some biometric identification techniques (voice or face recognition techniques for example). This aspect and possible improvements of developed system are currently being investigated in the continuation of this work.

Technique used	Feature set = 3 AR coefficients			Feature set = 3 AR coefficients + error power		
	N	A R	VAR	N	A R	VAR
LDA	0.39	0.59	0.37	0.61	0.65	0.47
QDA	0.41	0.62	0.42	0.64	0.71	0.54

TABLE 2: Recognition rate (%) obtained in classification of signal sections corresponding to individual patients.

4. CONCLUSIONS

In this work, a method for automatic classification of ECG signals from three different groups – normal, arrhythmia and ventricular arrhythmia has been proposed. Method first uses smoothness priors approach to pre-process all ECG signals from the database in order to reduce the baseline drifts and other trends in the signals. Filterbank based method to detect peaks in the pre-processed ECG signals is applied in the second stage of the proposed system before the coefficients of an AR signal model are extracted and used to classify each section of ECG signal into one of three possible groups. Decision about the type of each ECG signal from the test set is then made depending on the group into which most of the sections from the same ECG have been clustered. Extracted features, parameter sets are well separated in feature space and accurately classified, indicating that the high classification accuracy can be expected in the practical application of the proposed system. For the standard set of ECG test signals available at MIT/BIH and Politecnico of Milano web sites developed method has managed to achieve 100% accurate classification of three heart conditions.

In addition to this task, as a first step towards possible ECG based patient identification similar approach has been used to decide which of the analysed signal sections belong to the same ECG signal. To enhance the system performance in this application of the system, feature set for this task was extended with one more parameter – power of AR modeling error. Initial results indicate the significant increase in the recognition rate when extended feature set is used and demonstrate the potential of proposed approach for ECG based person recognition task.

Further work to enhance the patient recognition capabilities and accuracy of the system is ongoing and is currently focusing on the use of more complex signal models (ARMA or ARIMA) with the aim of extracting additional signal features needed for more accurate patient recognition. Use of other, advanced classification algorithms will also be considered in the continuation of this study.

5. REFERENCES

- [1] B. Surawicz. *Electrophysiologic Basis of E.C.G. and Cardiac Arrhythmias*, Lippincott Williams and Wilkins illustrated edition, 1995.
- [2] J. Adamec and R. Adamec. *ECG Holter: Guide to Electrocardiographic Interpretation*, Springer, 2008.
- [3] B. Kohler, C. Hennig and R. Orglmeister. "The principles of software QRS detection", *IEEE Eng Med Biol Mag*, vol 21, issue 1, pp. 42-57, 2002.
- [4] J. Pan and W. Tompkins. "A Real Time QRS Detection Algorithm", in *Biomedical Engineering IEEE Transactions on*, vol. 32, issue 3, pp. 230-236, 1985.
- [5] I. Romero and L. Serrano. "ECG frequency domain features extraction: a new characteristic for arrhythmias classification", in *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*. vol. 2, pp. 2006-2008 2001.
- [6] A. R.M.S.S. and B. Boashash. "Time-frequency domain features of ECG signals: their application in P wave detection using the cross Wigner-Ville distribution. in Acoustics, Speech, and Signal Processing", *Acoustics, Speech, and Signal Processing. ICASSP-89.,1989 International Conference on*, vol. 3, pp. 1524 - 1527, 1989.
- [7] L. Senhadji. "Comparing wavelet transforms for recognizing cardiac patterns", *Engineering in Medicine and Biology Magazine,IEEE*, vol. 14, issue 2, pp. 167-173, 1995.
- [8] C. Zheng and C. Tai. "Detection of ECG characteristic points using wavelet transforms", *IEEE Trans Biomed Eng*, vol. 42, issue 1, pp. 21-8, 1995.
- [9] Y. Hu, S. Palreddy and ,W. Tompkins. "A patient-adaptable ECG beat classifier using a mixture of experts approach", *IEEE Trans Biomed Eng*, vol. 44, issue 9, pp. 891-900, 1997.
- [10] M. Lagerholmet. "Clustering ECG complexes using Hermite functions and self-organizing maps", *Biomedical Engineering, IEEE Transactions on*, vol. 47, issue 7, pp. 838-848, 2000.
- [11] D. Cuesta-Frau. "Feature extraction methods applied to the clustering of electrocardiographic signals. A comparative study. in Pattern Recognition", in *Proceedings.16th International Conference on*. vol.3, pp. 961-964, 2002.
- [12] M. a. Nygård and L. Sörnmo. "Delineation of the QRS complex using the envelope of the e.c.g", *Medical and Biological Engineering and Computing*, vol. 21, issue 5, pp. 538-547, 1983.
- [13] P. e. a. Laguna. " Adaptive estimation of QRS complex wave features of ECG signal by the hermite model", *Medical and Biological Engineering and Computing*, vol. 34, issue 1, pp. 58-68, 1996.
- [14] G. Dayong. "Bayesian ANN classifier for ECG arrhythmia diagnostic system: a comparison study. in Neural Networks", *Proceedings. IEEE International Joint Conference on*. vol.4, pp. 2383-2388. 2005.

- [15] J. Kors and J. van Bommel. "Methods Inf Med, Classification methods for computerized interpretation of the electrocardiogram", *Methods of information in Medicine on*, vol. 29, issue 4, pp. 330-6, 1990.
- [16] D. Coast. "An approach to cardiac arrhythmia analysis using hidden Markov models. Biomedical Engineering", *IEEE Transactions on*, vol. 37, issue 9, pp. 826-836, 1990.
- [17] T. Yeap, F. Johnson and M. Rachniowski. "ECG Beat Classification By A Neural Network", *in Engineering in Medicine and Biology Society 1990, Proceedings of the Twelfth Annual International Conference of the IEEE*, pp. 1457-1458, 1990.
- [18] S. Osowski and L. Tran Hoai. "ECG beat recognition using fuzzy hybrid neural network", *Biomedical Engineering, IEEE Transactions on*, vol. 48, issue 11, pp. 1265-1271, 2001.
- [19] W. Yang. "A short-time multifractal approach for arrhythmia detection based on fuzzy neural network", *Biomedical Engineering, IEEE Transactions on*, vol. 48, issue 9, pp. 989-995, 2001.
- [20] R. Silipo and C. Marchesi. "Artificial neural networks for automatic ECG analysis", *Signal Processing, IEEE Transactions on*, vol. 46, issue 5, pp. 1417-1425, 1998.
- [21] P. Domingos and M. Pazzani. "Machine Learning, On the Optimality of the Simple Bayesian Classifier under Zero-One Loss", *Machine learning*, vol. 29, issue 2, pp. 103-130, 1997.
- [22] M. Tjoa. "Artificial neural networks for the classification of cardiac patient states using ECG and blood pressure data" *in Intelligent Information Systems Conference, The Seventh Australian and New Zealand 2001*, pp.323-327, 2001.
- [23] G. Dingfei, S. Narayanan and M. K. Shankar. "Cardiac arrhythmia classification using autoregressive modeling", *BioMedical Engineering OnLine*, 2002. [Online]. USENET: <http://www.biomedical-engineering-online.com/content/1/1/5>., Nov. 13, 2002 [Aug. 28, 2013].
- [24] J. Huang, Z. Jianping and X. Yiqing. " Source classification using pole method of AR model", *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-97*., 1997, vol. 1, pp. 567 - 570, 1997.
- [25] A. Ouelli, B. ElhadadiL, H. Aissaoui and B. Bouikhalene. "AR Modeling for Automatic Cardiac Arrhythmia Diagnosis using QDF Based Algorithm", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, Issue 5, 2012.
- [26] D. Gari, A. Francisco and M. Patrick. *Biomedical Engineering. :Advanced Methods And Tools for ECG Data Analysis*, Artech House, Inc., 2006.
- [27] M. Tarvainen, P. Ranta-aho and P. Karjalai. "An advanced detrending method with application to HRV analysis", *Biomedical Engineering, IEEE Transactions on*, vol. 49, issue 2, pp. 172-175, 2002.
- [28] J. F. Moraes. "A QRS complex detection algorithm using electrocardiogram leads", *Computers in Cardiology*, pp. 205 - 208, 2002.
- [29] P. Q. Liu. "A robust method for QRS detection based on modified p-spectrum. Acoustics, Speech and Signal Processing", *IEEE International Conference on 2008*, pp. 501 – 504,

2008.

- [30] V. Afonso. "ECG beat detection using filter banks", *Biomedical Engineering, IEEE Transactions on*, vol. 46, issue 2, pp. 192-202, 1999.
- [31] S. Marple. *Digital spectral analysis: with applications*. Прентике-Халл сигнал процессинг сериес, Prentice-Hall., 1987.
- [32] M. Hayes. *Statistical digital signal processing and modeling*, John Wiley & Sons., 1996.
- [33] M. De Hoon. "Why Yule-Walker should not be used for autoregressive modelling", *Annals of Nuclear Energy*, vol. 23, issue 15, pp. 1219-1228, 1996.
- [34] K. Burnham and D. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*, Springer., 2002.
- [35] J. Shadbolt and J. Taylor. *Perspectives in neural computing : Neural networks and the financial markets: predicting, combining, and portfolio optimisation*, Springer., 2002.
- [36] A. Un-Chi Yeh and W.-J. Wang. "Heartbeat Case Determination Using Fuzzy Logic Method on ECG Signals", *International Journal of Fuzzy Systems*, vol. 11, issue 4, 2009.
- [37] M. Tavassoli, M. Ebadzadeh and H. Malek. "Classification of cardiac arrhythmia with respect to ECG and HRV signal by genetic programming", *Canadian Journal on Artificial Intelligence, Machine Learning and Pattern Recognition*, vol. 3, 2012.
- [38] Łeski and A. Momot. "Bayesian and empirical Bayesian approach to weighted averaging of ECG signal", *Bulletin of the polish academy of sciences*, vol. 55, Issue 4, 2007.
- [39] D. P. S.Karpagachelvi. "ECG Feature Extraction Techniques - A Survey Approach", (*IJCSIS*) *International Journal of Computer Science and Information Security*, vol. 8, issue 1, 2010.
- [40] F. Van der Heijden. *Classification, Parameter Estimation and State Estimation - An Engineering Approach Using MATLAB*, John Wiley & Sons., 2004.
- [41] G. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition (Wiley Series in Probability and Statistics)*, Wiley-Interscience., 2004.
- [42] D. Milano. "Politecnico Biosignals Archives on CD-ROM", (Copyright (C) Politecnico. 1992),, 1992.
- [43] G. Moody and R. Mark. "The MIT-BIH Arrhythmia Database on CD-ROM and software for use with it", in *Computers in Cardiology 1990, Proceedings.*, 1990.

Face Images Database Indexing for Person Identification Problem

Jyotirmay Dewangan

*M. Tech Student, School of Information Technology
Indian Institute of Technology Kharagpur
Kharagpur, 721302, India*

jyotirmay.sg@gmail.com

Somnath Dey

*Faculty, Department of Computer Science and Engineering
Indian Institute of Technology Indore
Indore, 453441, India*

somnathd@iiti.ac.in

Debasis Samanta

*Faculty, School of Information Technology
Indian Institute of Technology Kharagpur
Kharagpur, 721302, India*

dsamanta@iitkgp.ac.in

Abstract

Face biometric data are with high dimensional features and hence, traditional searching techniques are not applicable to retrieve them. As a consequence, it is an issue to identify a person with face data from a large pool of face database in real-time. This paper addresses this issue and proposes an indexing technique to narrow down the search space. We create a two level index space based on the SURF key points and divide the index space into a number of cells. We define a set of hash functions to store the SURF descriptors of a face image into the cell. The SURF descriptors within an index cell are stored into kd-tree. A candidate set is retrieved from the index space by applying the same hash functions on the query key points and kd-tree based nearest neighbor searching. Finally, we rank the retrieved candidates according to their occurrences. We have done our experiment with three popular face databases namely, FERET, FRGC and CalTech face databases and achieved 95.57%, 97.00% and 92.31% hit rate with 7.90%, 12.55% and 23.72% penetration rate for FERET, FRGC and CalTech databases, respectively. The hit rate increases to 97.78%, 99.36% and 100% for FERET, FRGC and CalTech databases, respectively when we consider top fifty ranks. Further, in our proposed approach, it is possible to retrieve a set of face templates similar with query template in the order of milliseconds. From the experimental results we can substantiate that application of indexing using hash function on SURF key points is effective for fast and accurate face image retrieval.

Keywords: Biometric, Face Identification, Biometric-data Indexing, SURF, Index Key Generation.

1. INTRODUCTION

Of late, biometric-based person authentication system is gaining importance due to its wide spread applications such as personal identification [1], PDA, smart card [2, 3], access control [4, 5], surveillance [6], forensic applications [1, 7, 8], biometric passports [5, 9, 10], national identity card registration [11, 12, 13] and human computer interaction, etc. [1]. In these applications, there is a need to deal with large-scale databases [11, 14]. For example, Unique Identification Authority of India [11, 14] has planned to register 600 million users in India in next few years where the number of accesses per day (in different public and private domains) are expected to be around 1 to 5 million.

The major concern in the face identification is high dimensional feature vector. In a large-scale biometric system, exhaustive searching in face database to retrieve an identity is typically slow

and may not be acceptable. Also, the false acceptance error grows with the size of database [15]. As a consequence, the response time, search and retrieval efficiency are affected in addition to the accuracy of the system.

However, the current state-of-the-art research [24, 25, 26] in face recognition is mainly focused to solve the problem of variable lighting, pose, facial expression and aging in verification mode. The Face Recognition Grand Challenge (FRGC) [44, 45] shows a large improvement on face recognition accuracy for large number of face images with different lighting conditions, illuminations, poses and expressions. There are very few works [28, 29, 30] which addressed the problem of identification of a face image in large database. Though, some work [16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26] tried to alleviate the limitations of identification by proposing the dimensionality reduction techniques. But they use linear searching or nearest neighbor searching with the reduced low dimensional feature vector. Such techniques are not well suited for large scale applications. In this work, we address this problem by reducing the search space. We investigate for a fast and accurate mechanism to index the face databases.

In this paper, we propose an indexing technique to narrow down the search space. First, we preprocess the face image to detect face part from the background and enhance the intensity values of the extracted face image. We detect Speed Up Robust Feature (SURF) key points [27, 35] in different scale spaces of the preprocessed face images and extract SURF feature descriptors at each key point. We generate a set of sixty eight dimensional index keys from the key points, feature descriptors and identity of a face image. Among these the first four dimensions of index keys are used to create a two-level index space. The first level index space divides all face images into two groups depending on the value of the first dimension. In the second level index space, we create two index cubes based on the other three dimensions. Each cell of an index cube keeps the reference of a kd-tree where we store the feature descriptors and the identity of a set of face images. We apply a set of hash functions on the index keys to find the cell positions for face images. At the time of identification, we apply the same hash functions on the query key points and search the kd-tree to retrieve a small set of similar identities from the index space. Finally, we rank all retrieved identities according to their occurrences.

The rest of the paper is organized as follows. In Section 2, we discuss the existing face indexing techniques for face biometric identification system. We briefly describe the preprocessing task in Section 3. In Section 4, we discuss our proposed approach for face indexing. The experimental results are presented in Section 5. Finally, the paper concludes in Section 6.

2. RELATED WORK

In the existing literature, very few work have been reported for face indexing to reduce the search space. We describe these techniques [28, 29, 30] in this section. Lin et al. [28] proposed an indexing structure to search the face from a large database. They compute a set of eigenfaces based on the faces in the database. Then, they assign a rank to each face in the database according to its projection onto each of the eigenface. Similarly, they compute the eigenfaces for a query and rank a query face. They select a set of faces from the database corresponding to the nearest faces in the ranked position with respect to each eigenface of the query face. These selected faces are used for recognition.

Mohanty et al. [29] propose a linear subspace approximation method for face indexing. They build a linear model to create a subspace-based on the match scores. They apply a linear transformation to project face images into the linear subspace. To do this, first, they apply a rigid transformation obtained through principal component analysis and then a non-rigid, affine transformation. They use an iterative stress minimization algorithm to obtain a distance matrix in a low-dimensional space and propose a linear out-of-sample projection scheme for test images. Any new face image is projected into this embedded space using an affine transformation.

Kaushik et al. [30] propose a modified geometric hashing technique to index the face database. They extract features from a face image using Speeded-Up Robust Features (SURF) operator.

They apply mean centering, principal component analysis, rotation and normalization to preprocess the SURF features. Finally, they use geometric hashing to hash these features to index each facial image in the database.

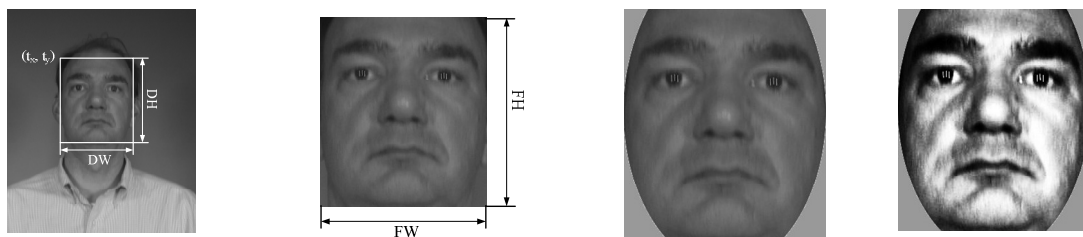
The major concern in eigenface and linear approximation method is the number of computations at the time of selecting top matches from the database. The number of computations in eigenface based method [28] depends on the number of eigenfaces considered for matching. To achieve good recognition rate a large number of eigenfaces are needed to consider and this does not give much computational advantages over the linear methods. The Linear subspace approximation method [29] does not give computational advantages when linear projection on raw templates (such as PCA, LDA etc.) are used for matching because the computation time for matching two face images in the database and mapping of face images into linear model space are same. The main concern in the geometric hashing technique is the selection of basis points for hashing [31, 32]. Wrong selection of basis points may deteriorate the performance of the geometric hashing based face indexing. Another problem in geometric hashing technique is that more number of basis points selection leads to the more computation time in indexing technique.

3. PREPROCESSING

Input face image of a biometric system contains background and is not necessarily good quality. To extract the features from a face image we need to enhance the image. This makes feature extraction task easy and ensures the quality of the extracted features. The steps followed in the preprocessing are briefly described in the following.

3.1 Geometric Normalization

The input face images may not be in same size and align in the same direction (due to movement of head at the time of capturing). We follow geometric normalization process of Bolme et al. [33] and Beveridge et al. [53] to align and scale the images so that the face images are in the same position, same size and at same orientation. To get the geometric normalized image, first, we rotate the face image by an angle such that the eye coordinates are in same line with the horizontal axis. After rotating the face image, we detect the face part from the rotated image. To do this we use Viola & Jones [34] face detection algorithm. The detected face is shown in Fig. 1(a). To make the face image scale invariant, we map the detected face part ($DW \times DH$) into a fixed size image ($FW \times FH$) by applying scaling transformation. Figure 1(b) shows the fixed size image of width FW and height FH .



(a) Detected face boundary (b) Detected face image mapped into fixed size image (c) Masked face image (d) Enhanced face image after masking

FIGURE 1: Face Images after Different Preprocessing Tasks.

3.2 Face Masking

We apply masking to separate the foreground region from the background region of a face image. The foreground region is corresponding to the clear face area which is the area of interest. This area contains the significant feature values. We mask the face image to ensure that the face recognition system does not respond to features corresponding to background, hair, clothing etc. We use an elliptical mask [33, 53] such that only the face from forehead to chin and left cheek to

right cheek is visible. Figure 1(c) shows the face image after applying the elliptical mask on the geometric normalized image where only face part is present in the image and background is masked out.

3.3 Intensity Enhancement

Intensity enhancement is required to reduce image variation due to lighting and sensor differences. We do the intensity enhancement in two steps [33]. First, we equalize the histogram of the unmasked face part and then normalize the intensity of the image to a mean of zero and standard deviation of one. Figure 1(d) shows the intensity enhancement image of the masked face image.

4. PROPOSED APPROACH

Speed-Up-Robust-Feature (SURF) [35, 27] method is known as a good image interest points (also called key points) and feature descriptors detector. We apply SURF method in our approach. We use SURF feature extraction method because it has several advantages over other feature extraction methods. The most important property of an interest point detector using SURF method is its repeatability. Repeatability means that the same interest points will be detected in different scales, orientations and illuminations of a given image. Another advantage is that the SURF method is computationally very fast. In addition to these, the SURF feature provides scale, rotation and illumination invariant feature descriptors.

Figure 2 shows the different tasks in enrollment and identification process of our approach. Both enrollment and identification processes have four common tasks as shown in Figure 2. In this section, first we present the key point extraction steps followed by orientation calculation of each key point. Then we discuss the feature descriptor extraction at each key point followed by the index key generation. Then, we describe index space creation and storing of index keys. Finally, we discuss the querying method to retrieve the identity from the index space.

4.1 Key Point Detection

We follow Bay et al. [27, 35] method to detect key points from an face image. The method consists of three steps.

In first step, we create scale spaces of the preprocessed image, which helps us to detect key points at different scales of the image. We construct eight distinct Gaussian filters with different sizes and different standard deviations. Then, we convolve the image with these Gaussian filters.

In next step, we calculate Hessian-matrix [27, 36] at each pixel position in the different scale space images. To compute the Hessian-matrix, we use integral images [27] to reduce the computation time of key point detection. We detect the key points based on the determinant values of the Hessian matrices. Note that all detected key points are not necessarily discriminant because the determinant of the Hessian Matrix does not produce local maximum or minimum

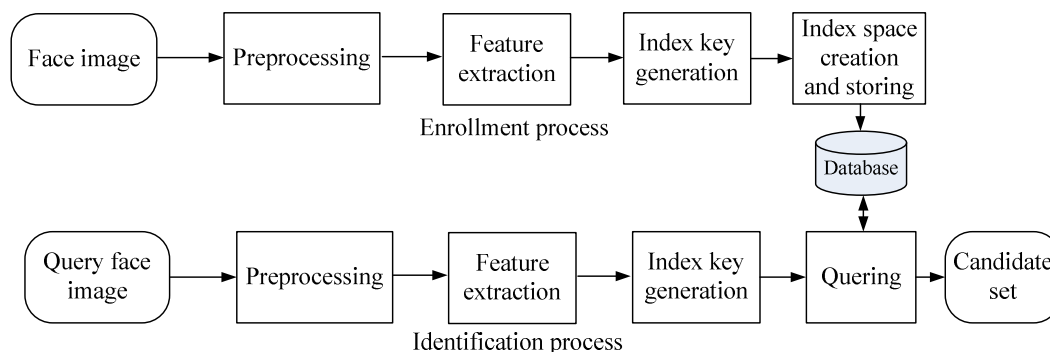


FIGURE 2: Overview of the Proposed Approach.

response at all detected points.

Finally, the most discriminant key points are localized from all the detected key points. To do this, first, we consider those points whose the determinant value is high and remove the points whose values are less than a threshold value. Bay et al. [35] shows that the threshold value 600 is good for detecting the discriminant key points from an image with average contrast and sharpness. Then, we perform non-maximal suppression to find the candidate key points. We do it by comparing each key point with its neighbors and finding the local maxima. We localize the key points by interpolating the maxima of the determinant of the Hessian matrix in scale and image spaces.

4.2 Orientation Assignment

We assign an orientation to a key point to extract rotation invariant features from the face image. The orientation is important because we extract the feature descriptors relative to this orientation in later stage. We follow Bay et al. [27, 35] method to compute the orientation at each key point. To find the orientation of a key point, first, we create a circular area centered with the key point. Then, we calculate the Haar wavelet responses [37] at each key point within the circular area in x and y direction and compute the weighted responses of the Haar wavelet responses with a Gaussian filter. The weighted response is represented by a point in the vector space. We find the dominating orientation at each key point by calculating the resultant vector in a window of size 60 degree. The longest vector leads as orientation of the key point.

A set of key points is detected from an image and we estimate orientation of each key point. A key point can be represented with position, orientation, scale space in which the key point is detected, Laplacian value and the determinant of Hessian matrix. Let k_1, k_2, \dots, k_L be the L detected key points of an input image. We represent the key points of an image as shown in Eq. (1).

$$\begin{aligned} k_1 &= x_1 \ y_1 \ \theta_1 \ \sigma_1 \ l_{s_1} \ h_{s_1} \\ k_2 &= x_2 \ y_2 \ \theta_2 \ \sigma_2 \ l_{s_2} \ h_{s_2} \\ &\dots \ \dots \\ k_L &= x_L \ y_L \ \theta_L \ \sigma_L \ l_{s_L} \ h_{s_L} \end{aligned} \tag{1}$$

In Eq. (1), (x, y) and θ represent the position and orientation of a key point, respectively; σ_i ($i=1,2, \dots, \delta$) denotes scale space at which key point is detected; l_s and h_s represent the Laplacian value and determinant of Hessian matrix, respectively.

4.3 Key Point Descriptor Extraction

In this step, we extract the feature descriptors at each key point from the scale space images as follows. Scale space images are created by applying Gaussian filter on the images (as discussed in Section 4.1). We follow SURF method Bay et al. [27] to extract the feature descriptors from the face image. First, we create a square window of size 20σ where σ is the scale or standard deviation of the Gaussian filter at which key point is detected. The window is centered at key point position and the direction of window is the same with the orientation of the key point (see Fig. 3). Now, the window is divided into 4×4 square sub regions and within each sub-region 25 (5×5) regularly distributed sample points are placed. We calculate Haar wavelet responses [37] at each sample point of a sub-region in x and y directions. We weight the Haar wavelet responses with a Gaussian filter with standard deviation 3.3σ centered at key point to reduce the effect of geometric deformations and localization errors. Let dx and dy be the Haar wavelet responses at each sample point within each sub-region. We consider $\sum dx$, $\sum |dx|$, $\sum dy$ and $\sum |dy|$ as four features at each sub-region. Hence, we create 64 ($4 \times 4 \times 4$) descriptors corresponding to each key point. In Eq. (2), $d_1, d_2, \dots, d_i, \dots, d_L$ represent descriptors of L key points and f_i^j represents the j^{th} descriptor of the i^{th} key point.

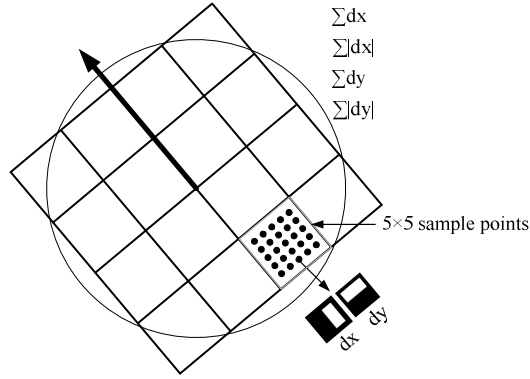


FIGURE 3: SURF Descriptor Extraction at a Key Point.

$$\begin{aligned}
 d_1 &= f_1^1 \ f_1^2 \ f_1^3 \ \dots \ f_1^j \ \dots \ f_1^{64} \\
 \dots & \dots \\
 d_i &= f_i^1 \ f_i^2 \ f_i^3 \ \dots \ f_i^j \ \dots \ f_i^{64} \\
 \dots & \dots \\
 d_L &= f_L^1 \ f_L^2 \ f_L^3 \ \dots \ f_L^j \ \dots \ f_L^{64}
 \end{aligned} \tag{2}$$

4.4 Index Key Generation

We extract all key points and feature descriptors from all face images. We can represent a face image with a set of index keys. The set of index keys are generated from the extracted key points of a face image such that for each key point there is an index key. More precisely, we use key point information, feature descriptors and identity of person as the constituents of an index key. We represent an index key as a row vector of sixty nine elements. The first four values of an index key contain the sign of Laplacian, position and orientation of a key point. Next sixty four values of the index key hold the feature descriptors corresponding to the key point and the last value keeps the identity of a person. The first four values are used to index the database and the feature descriptors are used to search the identity of a person. Let L_p be the number of key points (${}^p k_1, \dots, {}^p k_{L_p}$) and feature descriptors (${}^p d_1, \dots, {}^p d_{L_p}$) extracted from the p^{th} face image. Note that L_p may vary from one face image to another. Thus, L_p number of index keys are generated for the p^{th} face image. We represent the index keys of the p^{th} person in Eq. (3). The i^{th} index key (${}^p indx_i$) of the p^{th} face image is generated by the i^{th} key point (${}^p k_i$) and corresponding feature descriptors (${}^p d_i$), and the identity (id^p) of the p^{th} face image. In Eq. (3), ${}^p ls_i, {}^p x_i, {}^p y_i, {}^p \theta_i$ and ${}^p d_i$ represent the sign of Laplacian, x and y positions, orientation and feature descriptors of the i^{th} key point (${}^p k_i$), and id^p represents the identity of the p^{th} face image.

$$\begin{aligned}
 indx_1^p &= {}^p ls_1 \quad {}^p x_1 \quad {}^p y_1 \quad {}^p \theta_1 \quad {}^p d_1 \quad id^p \\
 indx_2^p &= {}^p ls_2 \quad {}^p x_2 \quad {}^p y_2 \quad {}^p \theta_2 \quad {}^p d_2 \quad id^p \\
 \dots & \dots \\
 indx_i^p &= {}^p ls_i \quad {}^p x_i \quad {}^p y_i \quad {}^p \theta_i \quad {}^p d_i \quad id^p \\
 \dots & \dots \\
 indx_{L_p}^p &= {}^p ls_{L_p} \quad {}^p x_{L_p} \quad {}^p y_{L_p} \quad {}^p \theta_{L_p} \quad {}^p d_{L_p} \quad id^p
 \end{aligned} \tag{3}$$

4.5 Index Space Creation and Storing

Once the index keys are generated, we have to create an index space to store all index keys into the database. The created index space helps us to find a match corresponding to a query in fast and accurate manner. To create index space, we use first four components of an index key. These are the sign of Laplacian (ls), positions (x and y) and orientation θ of a key point. All index keys can be classified into two groups based on the sign of the Laplacian value (ls) because this value distinguishes the brightness (dark and light) at a key point position. Note that all face images are aligned in the same direction and scaled to the same size in the preprocessing step. Hence, a key point will occur at the same or near to the same position in the image and the orientation of the key point will remain almost same although, the face images are captured at different time. So, we can divide the index keys in each group into sub-groups based on the positions (x and y) and orientation (θ) of a key point.

Due to the above characteristics of key points we propose a two-level index space to store the index keys. In the first level, we divide the index space based on the value of ls of index keys. The value of ls can be either '-1' for low intensity value or '+1' for high intensity value for a key point. Hence, the first level index space (LS) is divided into two sub-index spaces (LS_0 and LS_1) as shown in Fig. 4. In the second level, each sub-index space is divided into a number of cells based on the positions (x and y) and orientation (θ) of key points. We represent the second level index space ($INDX$) as a three dimensional index space. The three dimensions of index space are x , y and θ . Each dimension is in different scales. To bring each dimension in same scale we normalize the each dimension. To do this we quantize each dimension of the second level index space into the same number of units. Each dimension is quantized into δ number of units. The value of δ is decided experimentally (discussed in Section 5). We refer each three dimensional index space in the second level as an index cube. Each index cube contains δ^3 number of cells. Figure 4 shows two three-dimension index cubes for storing the index keys.

Now, we store all index keys into the index space based on the first four values of the index keys. Note that a number of index keys may map into a single cell of an index cube because index values of a set of index keys may fall within the same range. We find the cell positions for all index keys. To do this we define a set of hash functions based on the sign value of Laplacian, positions and orientation of a key point. Let ls , (x, y) and θ be the sign value of Laplacian, positions and orientation of a key point, respectively. Then the hash functions are defined in Eq. (4). In Eq. (4), ls' , x' , y' , and θ' are the cell index of the two-level index space and FH and FW

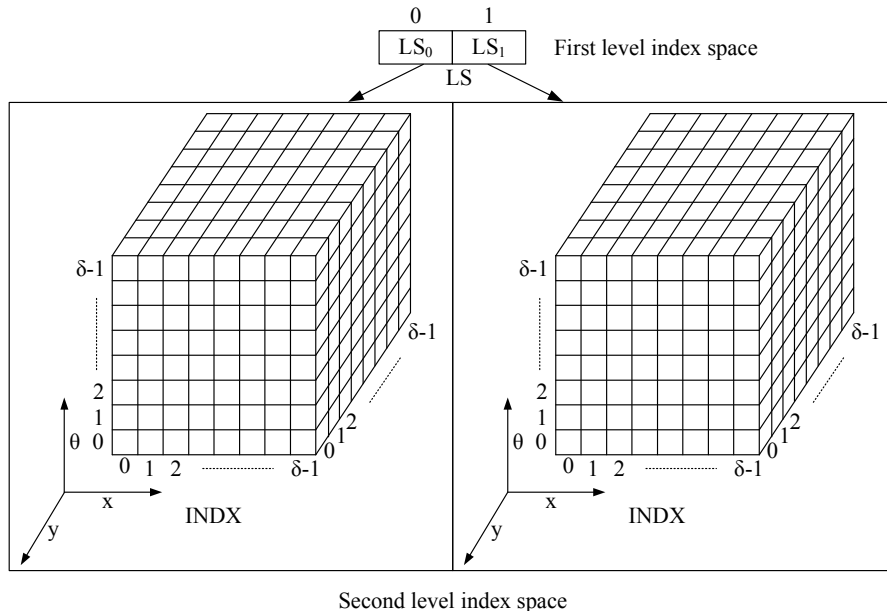


FIGURE 4: Proposed index space to store all index keys of all face images. represent the height and width of the normalized face image, respectively.

$$\begin{aligned}
 ls' &= H_{ls}(ls), \text{ where } H_{ls}(ls) = \frac{ls+1}{2} \\
 x' &= H_x(x), \text{ where } H_x(x) = \left\lfloor \frac{x \times \delta}{FW} \right\rfloor \\
 y' &= H_y(y), \text{ where } H_y(y) = \left\lfloor \frac{y \times \delta}{FH} \right\rfloor \\
 \theta' &= H_\theta(\theta), \text{ where } H_\theta(\theta) = \left\lfloor \frac{\theta \times \delta}{360} \right\rfloor
 \end{aligned} \tag{4}$$

We illustrate the storing of an index key into the proposed two-level index space with an example. Let $indx = \langle ls = -1, x=55, y=89, \theta=45, d_1, d_2, \dots, d_{64}, id \rangle$ be an index key of a face image of size 130×150 ($FW \times FH$), d_1 to d_{64} are the 64 dimensional feature descriptors of that key point and id be the identity of the subject. To store the feature descriptors (d_1, d_2, \dots, d_{64}) and the identity (id) into the index space, we apply the hash functions (defined in Eq. (4)) on ls, x, y and θ of the key point. The feature descriptors will be stored into the first index cube (LS_0) of the first level of index space because the value of ls' is 0 after applying the hash function on ls . The cell position of the index cube in second level index space ($INDX$) is decided by applying the hash function on x, y and θ . Let us assume that each dimension of second level index space is divided into 15 units (i.e. $\delta = 15$). After applying the hash function on x, y and θ the value of x', y' and θ' are 6, 8 and 1, respectively. Hence, the feature descriptors (d_1, d_2, \dots, d_{64}) and identity (id) of the index key ($indx$) is stored at $[6,8,1]$ location in the first index cube which is represented as $LS_0 \rightarrow INDX[6][8][1]$.

We may note that a cell of an index cube can contain a set of index keys. To store the index keys we propose two storing structures: linear storing structure and kd-tree storing structure. These storing structures are discussed in the following.

4.5.1 Linear Storing Structure

In this technique, we create a two-dimensional linear index space ($LNINDEX$) for each cell of the index cube. Each linear index space is assigned a unique id (lid) and this id is stored in the corresponding cell of the index cube. Note that there are δ^3 number of cells in each index cube. Hence, $2\delta^3$ number of linear index spaces is created to store all index keys using linear storage structure. The linear index space ($LNINDEX$) stores the 64-dimensional feature descriptors (d_1, d_2, \dots, d_{64}) of index keys and identities (id) of individuals. The size of the linear index space ($LNINDEX$) is $N \times (D+1)$ where N is the number of index keys in the cell and D is the number of feature descriptors within an index key. We store all index keys of a cell into the linear index space ($LNINDEX$) for that cell. Figure 5 shows the linear index space for a cell of an index cube. In Fig. 5, $LNINDEX_i$ is the linear index space for the i^{th} cell of the first index cube ($LS[0] \rightarrow INDX$). The cell stores the id (lid) of the linear index space ($LNINDEX_i$). The i^{th} cell of the index cube $LS[0] \rightarrow INDX$ is represented as $CELL[i]$. The method for creating linear index space is summarized in Algorithm 1. In Step 6 of Algorithm 1, we find the index cube from the first level index space. The cell position of the index cube is found in Step 7 to Step 9. Step 10 calculates the id of linear index space and Step 11 assigns that id to a cell of an index cube. We copy the descriptor values and the identities of index keys in Step 12 and 13.

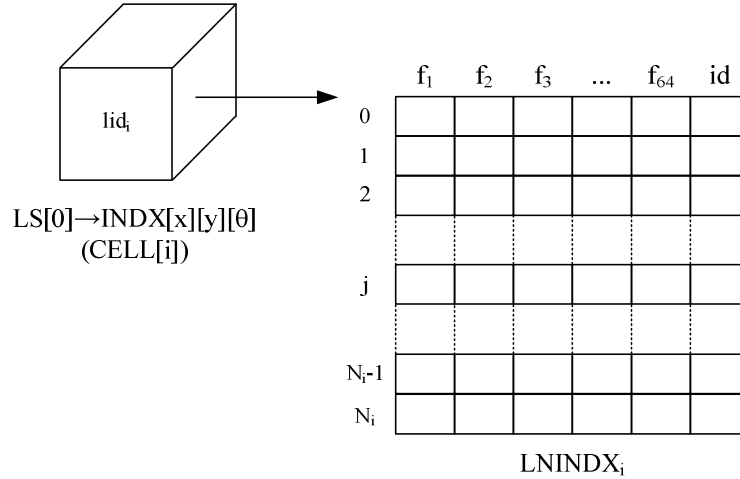


FIGURE 5: Linear indexing to store all index keys in i^{th} cell of the index space.

Algorithm 1 Creating index space with linear storing structure

Input: All index keys of all person's face image ($indx_1^p, indx_2^p, \dots, indx_{L_p}^p$; for $p=1$ to P). Two level index space ($LS[] \rightarrow INDX[][][]$).

Output: Index space ($LS[] \rightarrow INDX[][][]$) with linearly stored index keys for each cell ($CELL[] \rightarrow LNINDEX[][]$)

1. **for** $c=0$ to $2 \times \delta^3 - 1$ **do**
 2. $inc[c] = 0$ //Initialize linear index counter
 3. **end for**
 4. **for** $p=1$ to P **do**
 5. **for** $i=1$ to L_p **do**
 6. $ls = H_{ls}(ls_i^p)$ //Decide first level index space
 //Decide cell location of second level index space
 7. $x = H_x(x_i^p)$
 8. $y = H_y(y_i^p)$
 9. $\theta = H_\theta(\theta_i^p)$
 10. $lid = ls \times \delta^3 + x \times \delta^2 + y \times \delta + \theta$ //Calculate id for linear index space
 11. $LS[ls] \rightarrow INDX[x][y][\theta] = lid$ //Copy id of linear index space into a cell
 12. $CELL[lid] \rightarrow LNINDEX[inc[lid]] = {}^p d_i$ //Copy descriptor values of index key into linear index space
 13. $CELL[lid] \rightarrow LNINDEX[inc[lid]][65] = id^p$ //Copy identity of person into linear index space
 14. $inc[lid] = inc[lid] + 1$ //Increment linear index counter
 15. **end for**
 16. **end for**
-

4.5.2 Kd-tree Storing Structure

In this technique, we create a kd-tree for each cell of an index cube and assign a unique identity (kid) to the each kd-tree. The identity of the kd-tree is stored into the corresponding cell of the index cube. There are $2\delta^3$ number of cells in the index space. Hence, the total number of kd-trees

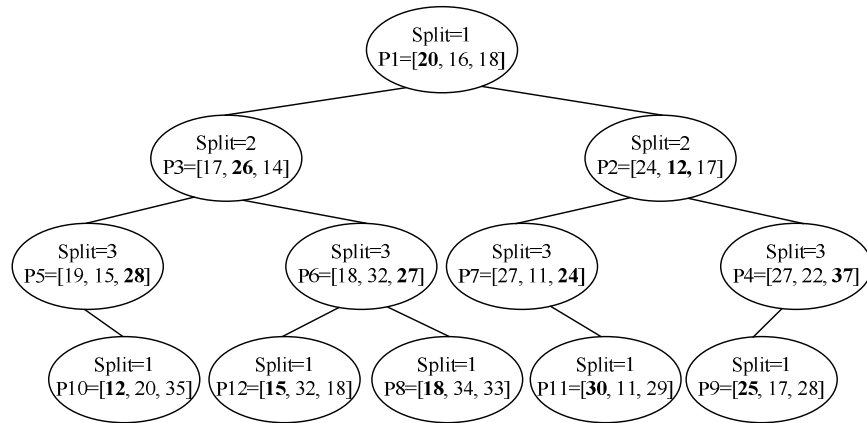
required is $2\delta^3$. All index keys of a cell are stored into a kd-tree. A kd-tree is a data structure for storing a finite set of points from a k-dimensional space [38]. The kd-tree is a binary tree in which every node stores a k-dimensional point. In other words, the node of a kd-tree stores an eight dimensional point. The node structure of kd-tree is shown in Fig. 6(a). In a node of kd-tree, *keyVector* field stores the k-dimensional index key and *Split* field stores the splitting dimension or a discriminator value. *leftTree* and *rightTree* store a kd-tree representing the pointers to the left and the right of the splitting plane, respectively. For an example, let there are eleven number of three dimensional points (P_1, P_2, \dots, P_{11}) as shown in Fig. 6(b) and *kd-tree* with these points is shown in Fig. 6(c). We insert the first point P_1 at the root of the *kd-tree*. At the time of insertion, we choose one of the dimensions as a basis (*Split*) of dividing the rest of the points. In this example, the value of *Split* in the root node is 1. In other words, if the value of the first dimension of the current point to be inserted is less than the same of the root, then the point is stored in *leftTree* otherwise in *rightTree*. This means all items to the left of root will have the first dimension value less than that of the root and all items to the right of the root will have greater than (or equal to) that of the root. The point P_2 is inserted in the right kd-tree and P_3 is inserted in the left *kd-tree* of the root. When we insert the point P_4 , we first compare the first dimension value of P_4 with the root and then compare the second dimension value of P_4 with the second dimension value of P_2 at next level. The point P_4 is inserted in the right *kd-tree* of the point P_2 . Similarly, we insert all other points into the kd-tree. First dimension will be chosen again as the basis ($Split=1$) for discrimination at level 3.

keyVector: k-dimensional point
Split: splitting dimension
leftTree: A kd-tree representing the points to the left of the splitting plane
rightTree: A kd-tree representing the points to the right of the splitting plane

- | | |
|-----------------|------------------|
| P1 = <20 16 18> | P7 = <27 11 24> |
| P2 = <24 12 17> | P8 = <18 34 33> |
| P3 = <17 26 14> | P9 = <25 17 28> |
| P4 = <27 22 37> | P10 = <12 20 35> |
| P5 = <19 15 28> | P11 = <30 11 29> |
| P6 = <18 32 27> | P12 = <15 32 18> |

(a) Components of a kd-tree node

(b) Sample 3D points



(c) Kd-tree example

FIGURE 6: Structure of a kd-tree and an example of kd-tree with 3-dimensional points.

We store all sixty four dimensional points (descriptors of an index key) within a cell in a kd-tree data structure. To store the index keys we apply the method proposed by Arya and Mount [54, 55] which follows Bentley [38] kd-tree insertion method. The maximum height of the optimized kd-tree with N number of k-dimensional point is $\lceil \log_2(N) \rceil$ [40]. The kd-tree structure for the i^{th} cell is shown in Fig. 7. In Fig. 7, $KDINDX_i$ is the kd-tree for the i^{th} cell of the first index cube ($LS[0] \rightarrow INDX$). The cell stores the identity (*kid*) of the kd-tree ($KDINDX_i$). The i^{th} cell of the first index cube $LS[0] \rightarrow INDX$ is represented as $CELL[i]$. We summarize the method for creating kd-

tree index space in Algorithm 2. Step 6 of Algorithm 2 finds the position of first level index space and Step 7 to Step 9 calculate the cell position in the index cube. In Step 10, we calculate the unique identity of the kd-tree. We assign the identity of the kd-tree into a cell of an index cube in Step 11. Finally, we insert the descriptor values and the identity of an individual into the kd-tree in Step 14.

4.6 Querying

Querying is the process of retrieving a set of candidates from the enrolled face templates (also called gallery) corresponds to a query template. The templates in the candidate set are most likely to match with the query face template. We investigate two different searching techniques to retrieve the face templates from the gallery of two different storage structures discussed previously. In each searching technique, first we generate the index keys corresponding to the query face using index key generation technique (discussed in Section 4.4). Let the index keys generated from a query face are represented in Eq. (5).

$$\begin{aligned}
 indx_1^q &= {}^q l_{s_1} \quad {}^q x_1 \quad {}^q y_1 \quad {}^q \theta_1 \quad {}^q d_1 \quad id^q \\
 \dots & \dots \dots \dots \dots \dots \dots \dots \\
 indx_i^q &= {}^q l_{s_i} \quad {}^q x_i \quad {}^q y_i \quad {}^q \theta_i \quad {}^q d_i \quad id^q \\
 \dots & \dots \dots \dots \dots \dots \dots \dots \\
 indx_{L_q}^q &= {}^q l_{s_{L_q}} \quad {}^q x_{L_q} \quad {}^q y_{L_q} \quad {}^q \theta_{L_q} \quad {}^q d_{L_q} \quad id^q
 \end{aligned} \tag{5}$$

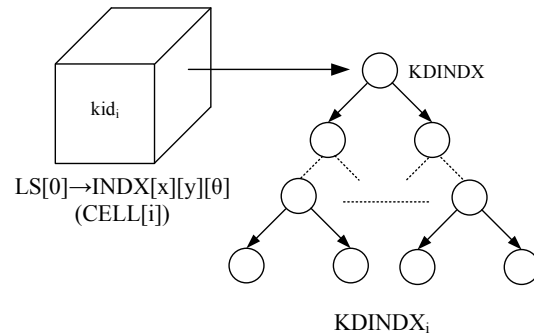


FIGURE 7: Kd-tree to store the index keys of i^{th} cell of the index space.

Algorithm 2 Creating index space with kd-tree storing structure}

Input: All index keys of all person's face image ($indx_1^p, indx_2^p, \dots, indx_{L_p}^p$; for $p=1$ to P). Two level index space ($LS[] \rightarrow INDX[][][]$).

Output: Index space ($LS[] \rightarrow INDX[][][]$) with kd tree stored index keys for each cell ($CELL[] \rightarrow KDINDEX$).

1. **for** $c=0$ **to** $2 \times \delta^3 - 1$ **do**
2. $inc[c] = 0$ //Initialize linear index counter
3. **end for**
4. **for** $p=1$ **to** P **do**
5. **for** $i=1$ **to** L_p **do**
6. $ls = H_{ls}(ls_i^p)$ //Decide first level index space
 //Decide cell location of second level index space
7. $x = H_x(x_i^p)$

8. $y = H_y(y_i^p)$
9. $\theta = H_\theta(\theta_i^p)$
10. $kid = ls \times \delta^3 + x \times \delta^2 + y \times \delta + \theta$ //Calculate id for kd-tree index space
11. $LS[ls] \rightarrow INDX[x][y][\theta] = kid$ //Copy id of the kd-tree into a cell
12. $Temp[] = {}^p d_i$ //Copy descriptors of index key into temporary vector
13. $Temp[65] = id^p$ //Copy identity of person into the temporary vector
14. Insert $Temp$ into $Kd-tree_{kid}$
15. $inc[kid] = inc[kid] + 1$ //Increment kd-tree index counter
16. **end for**
17. **end for**

The i^{th} query index key is represented as $indx_i^q = {}^q ls_i, {}^q x_i, {}^q y_i, {}^q \theta_i, {}^q d_i, id^q$ where ${}^q ls_i, {}^q x_i, {}^q y_i$ and ${}^q \theta_i$ represent the sign of Laplacian, x and y position, and orientation of the i^{th} key point (${}^q k_i$), and ${}^q d_i$ and id^q represent the feature descriptor of the i^{th} key point and identity of the query face image, respectively. Then, we apply indexing on the first level index space using the value of ls of the query index key. The indexing is done using hash functions defined in Eq. (4). The first level of indexing selects the index cube for a query index key. Let us assume that the value of ls of the i^{th} index key of query is -1. Then index cube ($LS[0] \rightarrow INDX$) in the first level index space (LS) is selected for the i^{th} index key of the query. Then, we use the value of x, y and θ of the query index key to find the cell position of the index cube in the second level index space. The cell position is calculated using the hash functions defined in Eq. (4). Then, the candidate set is generated by counting the vote received for each identity of the retrieved index keys from the database. A candidate set CS is shown in Fig. 8. The id and $vote$ fields of the CS store the identity of an individual and the number of vote received for that identity. The candidate set is generated for every type of searching. To generate the candidate set we search the corresponding linear or kd-tree storage whose identity is stored in the cell of an index cube and find the closest match in the linear or kd-tree index space. If x, y and θ of the i^{th} index key of a query select the $LS[0] \rightarrow INDX[x][y][\theta]$ cell of the index cube ($LS[0] \rightarrow INDX$) and retrieve the i^{th} linear or kd-tree identity then we find the closest match in the $LNINDEX_i$ linear index space for linear search and $KDTREE_i$ for kd-tree search. Finally, ranks are calculated based on the vote received for each identity. The search techniques are described in the following.

4.6.1 Linear Search

In linear search, first we find the cell position in an index cube for a query index key. Then, we search the linear index space of that cell. We compute Euclidean distance between feature descriptor of a query index key and all the feature descriptors stored in the linear index space to find a match. Let the j^{th} cell in the index cube is selected for the i^{th} index key of a query. Then, we select the linear index space ($CELL[j] \rightarrow LNINDEX$) corresponding to the j^{th} cell to find a match. We compute the Euclidean distances between the feature descriptors of the i^{th} index key of the query and all the descriptors stored in the linear index space ($CELL[j] \rightarrow LNINDEX$) using Eq. (6). We retrieve the identity corresponding to the minimum distance. The retrieved identity is then placed in the candidate set (CS) and cast a vote for this identity. We follow the same procedure for all other index keys of the query face. We summarize the linear searching method in Algorithm 3. In Step 2 of Algorithm 3, we initialize the length of each linear index space. Step 6 and Steps 7 to 9 find the index of the first and second level index spaces, respectively. We calculate the cell id in Step 10. In Steps 11 to 19, we find the minimum distance for an index key of query face and retrieve the identity corresponding to the minimum distance. Steps 23 to 29 generate the candidate set for a query index key. Finally, we sort the candidate set in Step 33.

$$ED_{i,j} = EuclidDist(d_j, {}^q d_i),$$

$$\text{where, } EuclidDist(d_j, {}^q d_i) = \sum_{f=1}^{64} (d_j^f - {}^q d_i^f)^2 \quad (6)$$

4.6.2 Kd-tree Search

In kd-tree search, first we find the cell position in an index cube for an index key of a query. Then we retrieve the id of a kd-tree (*kid*) from the cell and search the kd-tree corresponding to the retrieved kd-tree id. We apply hash functions to find the cell position in the index cube. Let the j^{th} cell in the index cube is selected for the i^{th} query index key. Then we search kd-tree index space ($CELL[j] \rightarrow KDINDEX$) corresponding to the j^{th} cell to find a match. We apply approximate nearest neighbor search [54, 55, 56, 57] to reduce the searching time. Arya and Mount's [54, 56] approximate k nearest neighbor search method is used to search the kd-tree. In this technique, we examine only the k closest bins of the kd-tree and use a priority queue to identify the closest bins based their distances from query. The expected searching complexity of the nearest neighbor search can be reduced to $O(kd \log n)$ and space complexity is $O(dn)$. For this purpose, a public domain library (FLANN) [56, 57] for faster approximate nearest neighbors search is available. In our approach, we utilize this library for implementing kd-tree algorithms. We retrieve the identity corresponding to the closest match from the kd-tree. The retrieved identity is then placed in the candidate set (*CS*) and cast a vote for this identity. We follow the same procedure for all other index keys of the query face. The searching method for kd-tree index space is summarized in Algorithm 4. Step 3 and Steps 4 to 6 of Algorithm 4 find the index of the first and second level index spaces, respectively. In Step 7, we calculate the cell id of an index cube for a query index key. Step 8 finds the nearest neighbor for a query index key and Step 9 retrieves the identity corresponding to the nearest neighbor. In Steps 11 to 15, we generate the candidate set for a query index key. Finally, we sort the candidate set in Step 18.

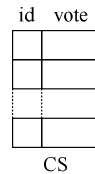


FIGURE 8: Schematic view of the candidate set.

Algorithm 3 Candidate set generation in linear search from index space

Input: All index keys from query face image $indx_1^q, indx_2^p, \dots, indx_{L_q}^q$, index space ($LS[] \rightarrow INDX[][][]$) with linearly stored index keys for each cell ($CELL[] \rightarrow LNINDEX[][]$).

Output: Candidate Set ($CS[] \rightarrow (id, vote)$)

1. **for** $cellid = 0$ **to** $2 \times \delta^3 - 1$ **do** //Initialize length in each linear index space
2. $KEYS[cellid] =$ Number of keys in $CELL[cellid] \rightarrow LNINDEX[][]$
3. **end for**
4. $idc = 1$
5. **for** $i = 1$ **to** L_q **do** // L_q is the total number of query index key
6. $ls' = H_{ls}(ls_i^q)$ //Find first level index space
 //Find cell location of second level index space
7. $x' = H_x(x_i^q)$
8. $y' = H_y(y_i^q)$
9. $\theta' = H_\theta(\theta_i^q)$
10. $cellid = LS[ls'] \rightarrow INDX[x'][y'][\theta']$ //Calculate cell id of an index cube
 //Retrieve the matched identities from $CELL[cellid] \rightarrow LNINDEX[][]$
11. $MinDist = \infty$
12. **for** $j = 0$ **to** $KEYS[cellid] - 1$ **do** // $KEYS[cellid]$ is the total number of index key in the $cellid^{th}$ cell
13. $ED_{i,j} = EucladDist(CELL[cellid] \rightarrow LNINDEX[j], qd_i)$


```

14.   if  $ED_{i,j} \leq MinDist$  then //Find match identities corresponding to the minimum distance
15.        $m = 1$ 
16.        $MatchId[m] = CELL[cellid] \rightarrow LNINDEX[j][65]$ 
17.   else if  $ED_{i,j} = MinDist$  then
18.        $m = m + 1$ 
19.        $MatchId[m] = CELL[cellid] \rightarrow LNINDEX[j][65]$ 
20.   end if
21. end for
22. for  $j = 1$  to  $m$  do
23.      $id = MatchId[m]$ 
24.     if  $id \notin CS[ ] \rightarrow (id)$  then //Generate candidate set
25.        $CS[idc] \rightarrow id = id$ 
26.        $CS[idc] \rightarrow vote = 1$ 
27.        $idc = idc + 1$ 
28.     else
29.        $CS[id] \rightarrow vote = CS[id] \rightarrow vote + 1$ 
30.     end if
31.   end for
32. end for
33. Sort  $CS[ ] \rightarrow (id, vote)$  in descending order based on vote

```

Algorithm 4 Candidate set generation in kd-tree search from index space

Input: All index keys from query face image $indx_1^q, indx_2^p, \dots, indx_{L_q}^q$, index space ($LS[] \rightarrow INDX[][][]$) with kd-tree index space for each cell ($CELL[] \rightarrow KDINDEX[]$).

Output: Candidate Set ($CS[] \rightarrow (id, vote)$)

```

1.   $idc = 1$ 
2.  for  $i = 1$  to  $L_q$  do //  $L_q$  is the total number of query index key
3.     $ls' = H_{ls}(ls_i^q)$  //Find first level index space
    //Find cell location of second level index space
4.     $x' = H_x(x_i^q)$ 
5.     $y' = H_y(y_i^q)$ 
6.     $\theta' = H_\theta(\theta_i^q)$ 
7.     $cellid = LS[ls] \rightarrow INDX[x'][y'][\theta']$  //Calculate cell id of an index cube
    //Retrieve the matched identities from Kd-tree ( $Kd-tree_{cellid}$ ) by finding nearest
    neighbor of a query index key
8.     $NN = findNN(Kd-tree_{cellid},^q d_i)$  //Find the nearest neighbor
9.     $id = retrieveIdFromNN(NN)$  //Select id of nearest neighbor
10.  if  $id \notin CS[ ] \rightarrow (id)$  then //Generate candidate set
11.     $CS[idc] \rightarrow id = id$ 
12.     $CS[idc] \rightarrow vote = 1$ 
13.     $idc = idc + 1$ 
14.  else
15.     $CS[id] \rightarrow vote = CS[id] \rightarrow vote + 1$ 
16.  end if
17. end for
18. Sort  $CS[ ] \rightarrow (id, vote)$  in descending order based on vote

```

4.7 Analysis of the Proposed Approach

In this section, we analyze the time complexity of linear and kd-tree based search techniques in the proposed index space. We also analyze the memory requirement of the proposed method.

4.7.1 Searching Time Complexity

Let N be the total number of face images enrolled in the database and T_p be the average number of index keys in each enrolled face image. Thus, the total number of index keys in the index space is $T_n = T_p \times N$. If there are K number of cells in all index cubes, then the average number of index keys in each cell is $T_k = T_n/K$. Let T_q be the average number of index keys in each query face image. To perform linear search or kd-tree based search in the index space, first, we find the index cell position for a index key of a query using hash functions defined in Eq.(4). This operation requires $O(1)$ computation time for both type of searches. The time complexity analysis of linear and kd-tree based searches within the located cell are given in the following.

Linear Search: In linear search, $T_k \times T_q$ number of comparisons are required to retrieve a set of similar index keys and their identities, and $T_q \log T_q$ comparisons are required to sort the retrieved identities based on their ranks. Thus, we can calculate the average time complexity of linear search (denoted as T_{LS}) as follows.

$$\begin{aligned} T_{LS} &= O(1) \times T_q + T_k \times T_q + T_q \log T_q \\ &= O(1) \times T_q + \frac{N \times T_p}{K} \times T_q + T_q \log T_q \\ &= O(N) \end{aligned} \tag{7}$$

Kd-tree Search: The number of comparisons required in kd-tree based search to find a set of nearest index keys and their identities are $\log T_k \times T_q$, and to sort the retrieved identities based on their ranks are $T_q \log T_q$. Thus, we can calculate the average time complexity of kd-tree based search (denoted as T_{KS}) as follows.

$$\begin{aligned} T_{KS} &= O(1) \times T_q + \log T_k \times T_q + T_q \log T_q \\ &= O(1) \times T_q + \log \frac{N \times T_p}{K} \times T_q + T_q \log T_q \\ &= O(\log N) \end{aligned} \tag{8}$$

4.7.2 Memory Requirement

Let b_1 and b_2 bytes memory are required to store the reference of the index cubes into the first level index space and the reference of linear or kd-tree index spaces into the index cube, respectively. Let m bytes are required to store a feature value of an index key and 2 bytes are required to store an identity of an individual. If there are P individuals then we can compute the memory requirement for linear and kd-tree index spaces using Eq. (9) and (10), respectively. In Eq. (9) and (10), L_p represents the number of index keys for the p^{th} individual and δ denotes the number of quantization levels of the second level index space.

$$M_{LS} = 2 \times (b_1 + \delta^3 \times b_2) + \sum_{p=1}^P (64 \times m + 2) \times L_p \tag{9}$$

$$M_{KD} = 2 \times (b_1 + \delta^3 \times b_2) + \sum_{p=1}^P (64 \times m + 14) \times L_p \tag{10}$$

5. EXPERIMENTS AND EXPERIMENTAL RESULTS

This section describes the different experimental setups and the experimental results to evaluate the accuracy and the efficiency of our proposed approach.

5.1 Database

We perform our experiments on two widely used large face databases namely Color FERET [40, 41, 42] and FRGC V2.0 [43, 44, 45]. We also carry out our experiments on CalTech 256 [46, 47] face database. A detail description of each database is given in the following.

5.1.1 Color FERET Face Database

The FERET database is developed for the Facial Recognition Technology (FERET) program [41, 42]. The database is designed by the Defense Advanced Research Products Agency (DARPA) during 1993 to 1997 to give common standard for face recognition experiments. The database contains 11338 images from 994 different subjects. These images are collected in different sessions. The resolution of the captured images is 256×384 pixel. The database contains 2722 frontal images with different facial expressions (Neutral and Alternate). There are 1364 images with neutral expression and 1358 images with alternate expression. Figure 9(a) and (b) shows the four images with different facial expressions of two different subjects.

5.1.2 FRGC 2.0 Face Database

FRGC Still version 2.0 data set [43, 44, 45] is collected at University of Notre Dame as a part of Face Recognition Grand Challenge program. The primary goal of the FRGC program is to promote and advance the face recognition technology designed to support existing face recognition systems. This database contains color face images, which are taken in different lightning conditions and different environments. The database consists of 24038 frontal face images of 466 subjects. These images are captured in Fall 2003 and Spring 2004 semesters of 2003-2004 academic year. A total of 16024 images from all subjects are captured in indoor environment with two different protocols (FERET and Mugshot) and two different facial expressions (Neutral and Smiley) [43]. The resolution of each image is either 1704×2272 pixel or 1200×1600 pixel. The images are collected in 4007 subject sessions. Four images (FERET-Neutral, FERET-Smiley, Mugshot-Neutral and Mugshot-Smiley) are captured in each subject session. The database contains 4007 FERET-Neutral, 4007 FERET-Smiley, 4007 Mugshot-Neutral and 4007 Mugshot-Smiley face images. Figure 9(c) and (d) show four images with two facial expressions of two different subjects. FRGC Still version 2.0 data set [43, 44, 45] contains 8014 face images which are captured in outdoor environment with different backgrounds and different illuminations. Figure 9(e) shows two face images of two different subjects in different backgrounds.

5.1.3 CalTech 256 Face Database

Caltech-256 object category data set [47, 46] contains a total of 30607 images from 256 different categories. In our experiment, we use face category images of the Caltech-256 data set. The face category set consists of 432 face images from 28 subjects. Each face image is captured in complex background with different facial expressions. Figure 9(f) shows two face images of two different subjects from CalTech 256 face database.

5.2 Implementation Environment

All methods described in our approach are implemented using C programming language and OpenCV [48] image processing library on the Linux operating system. All methods are evaluated with Intel Core-2 Duo processor of speed 2.00GHz and 2GB RAM.

5.3 Performance Metrics

Accuracy and efficiency are the two main criteria usually considered to measure the performance of a face indexing technique. The accuracy of a face indexing approach is commonly evaluated

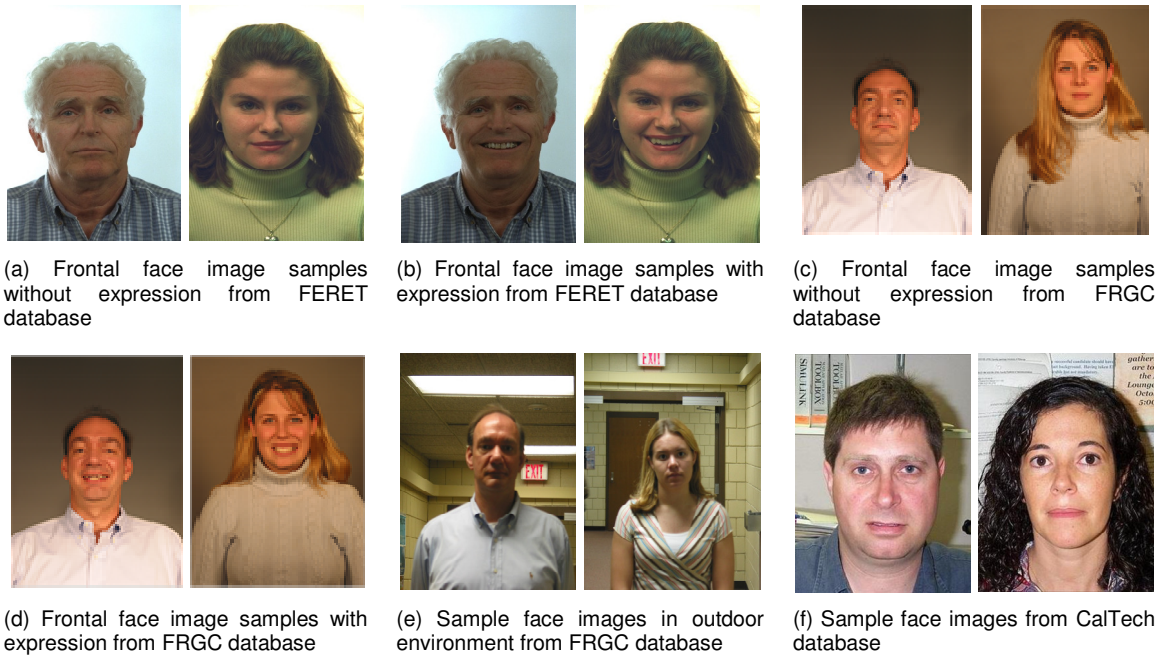


FIGURE 9: Sample images of FERET, FRGC and CalTech256 databases.

by the hit rate or recognition rate, and penetration rate [49, 50, 51, 52]. Hit rate is the percentage of probes for which the correct identities are retrieved within a top rank for a gallery by the indexing mechanism [50]. The penetration rate is the percentage of the database retrieved for a query face to get a correct match [50]. Let N_g be the number of entries in the database and N_p be the number of queries in the probe set. If N_r is the number of entries retrieved for the i^{th} probe then the penetration rate (PR) for a query is defined as in Eq. (11). If N_c ($N_c < N_p$) is the number of queries for which successful matches are found within the top r retrieved candidates then the hit rate (HR) at rank r is defined as Eq. (12)

$$PR = \frac{1}{N_p} \sum_{i=1}^{N_p} \frac{N_r}{N_g} \quad (11)$$

$$HR = \frac{N_c}{N_p} \quad (12)$$

We also substantiate our results in terms of cumulative match score. The cumulative match score gives the probability of at least one correct identity presents within a top rank which also represents the cumulative hit rate at different ranks. The cumulative hit rates at different ranks are represented with the Cumulative Match Characteristics (CMC) curve.

5.4 Evaluation Setup

To evaluate our proposed indexing method, we have partitioned each face database into two sets: *Gallery Set* and *Probe Set*. *Gallery Set* contains the face images which are enrolled into the index database and *Probe Set* contains the face images which are used as queries to search the index database. In our experiment, we create different gallery and probe sets for each database.

The description of different gallery and probe sets for FERET, FRGC and CalTech256 databases are given in Table 1.

5.5 Experiments and Results

We have conducted a number of experiments to evaluate the performance of our proposed indexing methods. The description of these experiments and the results of each experiment are given in the following.

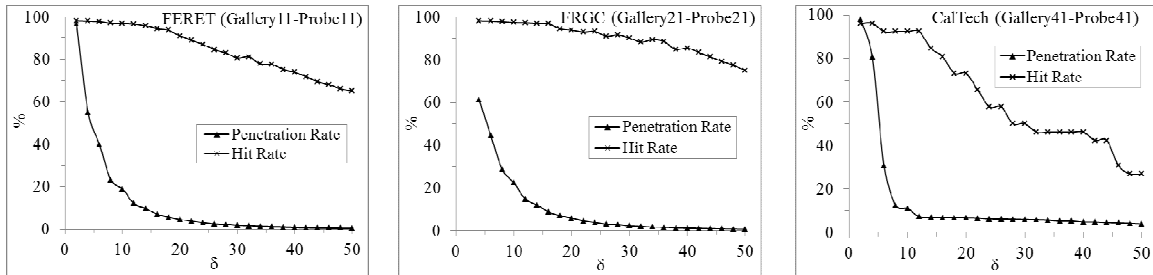
5.5.1 Experiment 1: Determining dimension quantization value of second level index space

To determine the value of the number of quantization (δ) of each dimension of the second level index space, we have done this experiment. We perform kd-tree based search for a set of query images with different values of δ . The value of δ is varied from 2 to 50 with increment of 2. This experiment is conducted with FERET, FRGC and CalTech databases. We use *Gallery11* and *Probe11* for FERET database, *Gallery21* and *Probe21* for FRGC database, and *Gallery41* and *Probe41* for CalTech database. The rank 1 hit rate and penetration rate for different values of δ are reported in Fig. 10. From Fig. 10(a), we observe that for FERET database rank 1 hit rate decreases nearly 2% when the value of δ is changed 2 to 14 whereas rank 1 hit rate decreases more than 4% when the value of δ is changed 16 to 20. On the other hand, the penetration rate decreases more than 85% when the value of δ is changed 2 to 14 but penetration rate decreases only 5% for the change of δ value 14 to 20 for FERET database. The same thing is observed from Fig. 10(b) for FRGC database also. But for Caltech database the value of δ equal to 12 gives better performance than the other values of δ (see Fig. 10(c)). Hence, in our other experiments, we choose the value of δ equal to 15 for FERET and FRGC databases, and 12 for Caltech database, respectively. However, user may choose the other values of δ according to their requirements.

DB	Name	# images	# subjects	Description
Color FERET	Gallery11	994	994	First face image with neutral facial expression of first session for all subjects.
	Gallery12	1984	992	First face image with neutral and alternate facial expressions of first session for all subjects.
	Probe11	992	992	First face image with alternate facial expression of first session for all subjects.
	Probe12	370	250	Face images with neutral facial expressions of other sessions for all subjects.
	Probe13	366	247	Face images with alternate facial expressions of other sessions for all subjects.
	Probe14	736	250	Face images with neutral and alternate facial expressions of other sessions for all subjects.
FRGC V2.0	Probe15	228	75	Face images with neutral and alternate facial expressions of other sessions for all subjects. But images are captured with minimum six months difference.
	Gallery21	466	466	First face image with neutral facial expression of first session for all subjects. Images are captured with FERET protocol.
	Gallery22	932	466	First face image with neutral and smiley facial expressions of first session for all subjects. Images are captured with FERET protocol.
	Probe21	466	466	First face image with smiley facial expression of first session for all subjects. Images are captured with FERET protocol.
	Probe22	3541	411	Face images with neutral facial expressions of other sessions for all subjects. Images are captured with FERET protocol
	Probe23	3541	411	Face images with smiley facial expressions of other sessions for all subjects. Images are captured with FERET protocol.
	Probe24	7082	411	Face images with neutral and smiley facial expressions of other sessions for all subjects. Images are captured with FERET protocol
Probe25	1134	193	Face images with neutral and smiley facial expressions of other sessions for all subjects. Images are captured with FERET protocol. The time difference from first	

			captured image is minimum six months.	
Probe26	466	466	First face image with neutral facial expression of first session for all subjects. Images are captured with Mugshot protocol.	
Probe27	466	466	First face image with smiley facial expression of first session for all subjects. Images are captured with Mugshot protocol.	
Probe28	3541	411	Face images with neutral facial expressions of other sessions for all subjects. Images are captured with Mugshot protocol	
Probe29	3541	411	Face images with smiley facial expressions of other sessions for all subjects. Images are captured with Mugshot protocol.	
Probe30	7082	411	Face images with neutral and smiley facial expressions of other sessions for all subjects. Images are captured with Mugshot protocol	
Probe31	1134	193	Face images with neutral and smiley facial expressions of other sessions for all subjects. Images are captured with Mugshot protocol. The time difference from first captured image is minimum six months.	
Probe32	8014	466	Face images with at outdoor environment.	
CalTech256	Gallery41	28	28	One face image from each subject. Face image is selected randomly.
	Gallery42	28	350	Eighty percent face images of each subject. Face images are selected randomly.
	Probe41	26	26	One face image from each subject. Face image is selected randomly from the rest of the Gallery1.
	Probe42	26	404	All face images of each subject except the Gallery 1 face images.
	Probe43	26	82	All face images of each subject except the Gallery 2 face images.

TABLE 1: Description of gallery and probe sets of FERET, FRGC and CalTech256 face databases.



(a) Hit rate and penetration rate with FERET Gallery11 and Probe11 sets for different values of δ

(b) Hit rate and penetration rate with FRGC Gallery21 and Probe21 sets for different values of δ

(c) Hit rate and penetration rate with CalTech256 Gallery41 and Probe41 sets for different values of δ

FIGURE 10: HR and PR with FERET, FRGC and CalTech256 databases for different values of δ

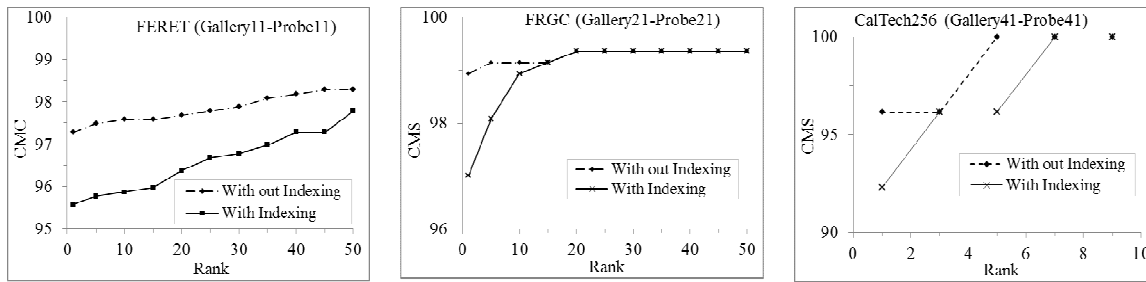
5.5.2 Experiment 2: Performance comparison without and with indexing

In this experiment, we compare the performance of the system with and without applying the proposed indexing technique. We use *Gallery11* and *Probe11* for FERET, *Gallery21* and *Probe21* for FRGC databases, and *Gallery41* and *Probe41* for CalTech databases. The CMC curves with and without indexing for different databases are shown in Fig. 11. Figure 11(a) shows that the approach without indexing gives better cumulative match score for FERET database. Whereas, From Fig. 11(b) and (c), we can see that the approach with and without indexing gives almost the same cumulative match score after 15th rank for FRGC database and after 7th rank for the CalTech database. We also report the rank 1 hit rate, penetration rate and average searching time for linear and kd-tree search using indexing and without indexing in Table 2. From Table 2, we can see that kd-tree search with indexing achieves 95.57%, 97% and 92.31% hit rate with 9.70%, 12.55% and 7.14% penetration rate for FERET, FRGC and CalTech databases,

Database	Performance	Linear		Kd-tree	
		Without Indexing	With Indexing	Without Indexing	With Indexing
FERET	Hit Rate	97.28	95.57	97.28	95.57
	Penetration Rate	100	10.54	49.43	7.90
	Average Search Time	4.46×10^5	114	2.21×10^5	85.40
FRGC	Hit Rate	98.93	97.00	98.93	97.00
	Penetration Rate	100	16.36	51.47	12.55
	Average Search Time	2.78×10^5	92.55	1.43×10^5	71.02
CalTech	Hit Rate	96.15	92.31	96.15	92.31
	Penetration Rate	100	26.91	61.78	23.72
	Average Search Time	1.24×10^4	8.10	7.64×10^3	7.14

TABLE 2: Comparison of the proposed approach with and without indexing using linear and kd-tree search.

respectively. We also observe that kd-tree search requires very less average searching time for all three databases.



(a) CMC curve with and without indexing for FERET Gallery11 and Probe11 sets

(b) CMC curve with and without indexing for FRGC Gallery21 and Probe21 sets

(c) CMC curve with and without indexing for CalTech256 Gallery41 and Probe41 sets

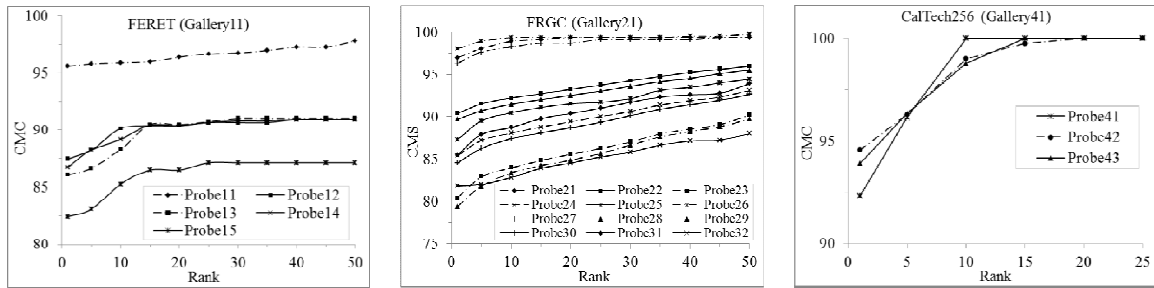
FIGURE 11: CMC curve with and without indexing for FERET, FRGC and CalTech256 databases.

5.5.3 Experiment 3: Performance with different probe sets

In this experiment, we check the performance of linear and kd-tree search with the proposed indexing method for different probe sets. The probe sets are created with different conditions as discussed in Table 1. We enroll the all images of *Gallery11* (with neutral expression), *Gallery21* (with neutral expression) and *Gallery41* into the database for FERET, FRGC and CalTech databases, respectively and use all probe sets to test the indexing performances of linear and kd-tree search. Figure 12(a) shows the CMC curve of all five probe sets of FERET database, Fig. 12(b) shows the CMC curve of all twelve probe sets of FRGC database and Fig. 12(c) shows the CMC curve of all three probe sets of CalTech database. From Fig. 12(a) and (b), we can note that cumulative match scores are reduced for the probe sets which contain the face images captured in more than six month difference. On the other hand, face images captured in different expressions but in the same session give the better results than the others. We observe that face images with complex background (*Probe32* of FRGC database) gives less match score than the others. However, face images with complex background for CalTech database give above 90% match score. We also report the rank 1 hit rate, penetration rate and searching time for linear and kd-tree search for different probe sets in Table 3. We observe that the penetration rate and searching time for kd-tree based search are less for all probe sets.

5.5.4 Experiment 4: Performance of multiple enrolments of a subject into the index space

We have done the experiment to check the effect of multiple enrolments on the performance. In this experiment, we enroll all samples of *Gallery12* for FERET, *Gallery22* for FRGC and *Gallery42* for CalTech databases, and test with all probe sets of FERET, FRGC and CalTech databases. The CMC curves of all probe sets are shown in Fig. 13. From Fig. 13, we can see that 100% cumulative match score is achieved for *Probe11* and *Probe21* because the images in the *Probe11* and *Probe21* sets are also in the *Gallery12* and *Gallery22*, respectively. We observe that in multiple enrolments of a subject, cumulative match scores for other probe sets are increased than that of in the single enrolments. We have computed the penetration rate, rank 1 hit rate and searching time for linear and kd-tree based search with multiple enrolments. The results are summarized in Table 4. From this experiment we observe that better rank 1 hit rate is achieved without affecting the penetration rate. Though, higher searching time is required to search the database with multiple enrolments.



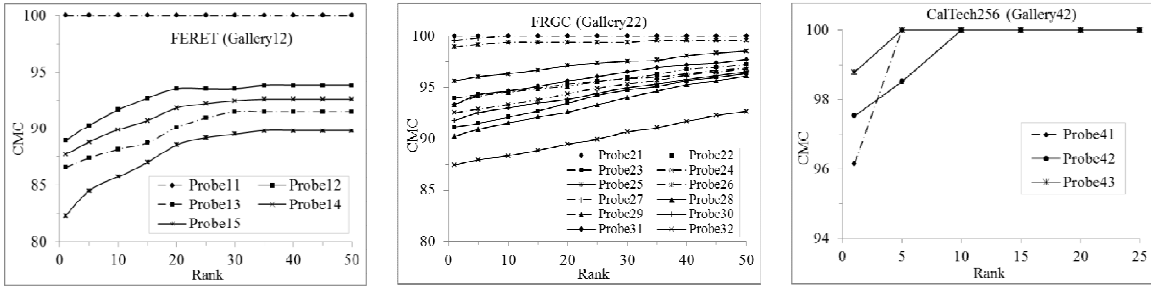
(a) CMC curves for different probe sets with FERET Gallery11 (b) CMC curves for different probe sets with FRGC Gallery21 (c) CMC curves for different probe sets with CalTech256 Gallery41

FIGURE 12: CMC curves for different probe sets with single enrolment of a subject with FERET, FRGC and CalTech256 databases

DB	Probe	Linear			Kd-tree		
		Hit rate	Penetration rate	Searching time (ms)	Hit rate	Penetration rate	Searching time (ms)
FERET	Probe11	95.57	10.54	114.00	95.57	7.90	85.40
	Probe12	87.44	11.13	121.98	87.44	8.34	91.38
	Probe13	86.11	11.28	122.32	86.11	8.45	91.64
	Probe14	86.75	11.20	121.29	86.75	8.39	90.87
	Probe15	82.44	10.62	114.29	82.44	7.96	85.62
FRGC	Probe21	97.00	16.36	92.55	97.00	12.55	71.02
	Probe22	90.40	14.66	82.48	90.40	11.25	63.29
	Probe23	80.41	16.13	92.01	80.41	12.38	70.60
	Probe24	85.40	15.40	87.66	85.40	11.81	67.27
	Probe25	87.31	15.69	88.48	87.31	12.04	67.89
	Probe26	98.07	14.62	81.73	98.07	11.22	62.71
	Probe27	96.36	16.20	91.73	96.36	12.43	70.39
	Probe28	89.70	14.41	81.72	89.70	11.06	62.71
	Probe29	79.39	16.07	91.24	79.39	12.33	70.01
	Probe30	84.54	15.24	84.86	84.54	11.69	65.12
	Probe31	85.46	15.42	86.88	85.46	11.83	66.67
	Probe32	81.40	15.47	86.38	81.81	11.87	66.28

CalTech	Probe41	92.31	26.91	8.19	92.31	23.72	7.22
	Probe42	94.55	27.52	8.29	94.55	24.27	7.31
	Probe43	93.90	27.72	8.48	93.90	24.44	7.47

TABLE 3: Performance of different probe sets with single enrolment of a subject in linear and kd-tree search.



(a) CMC curves for different probe sets with FERET Gallery12 (b) CMC curves for different probe sets with FRGC Gallery22 (c) CMC curves for different probe sets with CalTech256 Gallery42

FIGURE 13: CMC curves for different probe sets with multiple enrolment of a subject with FERET, FRGC and CalTech256 databases.

5.5.5 Experiment 5: False Match Rate (FMR) vs False Non Match Rate (FNMR)

We have analyzed the performance of our proposed system with respect to *false match rate (FMR)* and *false non match rate (FNMR)*. We use *Gallery11*, *Gallery21* and *Gallery42* datasets as gallery sets, and *Probe11*, *Probe21* and *Probe43* as probe sets for FERET, FRGC and CalTech databases, respectively. We have computed 992, 466 and 1297 genuine scores and 985056, 216690 and 27403 imposter scores for FERET, FRGC and CalTech databases, respectively. The receiver operating characteristics (ROC) curves show the trade-off between *FMR* and *FNMR* in Fig. 14. The equal error rates (ERR) of the system are 5.51%, 5.73% and 6.84% for FERET, FRGC and CalTech databases, respectively.

DB	Probe	Linear			Kd-tree		
		Hit rate	Penetration rate	Searching time(ms)	Hit rate	Penetration rate	Searching time(ms)
FERET	Probe11	100	9.33	207.66	100	6.74	149.98
	Probe12	88.94	9.85	215.93	88.94	7.11	155.96
	Probe13	86.56	9.99	216.59	86.56	7.21	156.43
	Probe14	87.73	9.92	214.91	87.73	7.16	155.22
	Probe15	82.31	9.40	205.35	82.31	6.79	148.31
FRGC	Probe21	100	13.69	164.07	100	10.14	121.54
	Probe22	91.11	12.27	146.87	91.11	9.09	108.79
	Probe23	93.93	13.50	163.63	93.93	10.00	121.21
	Probe24	92.52	12.88	153.78	92.52	9.54	113.91
	Probe25	95.59	13.13	157.14	95.59	9.73	116.40
	Probe26	98.93	12.23	148.34	98.93	9.06	109.88
	Probe27	99.57	13.56	160.79	99.57	10.04	119.11
	Probe28	90.23	12.06	144.72	90.23	8.93	107.20
	Probe29	93.31	13.45	160.29	93.31	9.96	118.73
Probe30	91.77	12.75	151.78	91.77	9.45	112.43	

	Probe31	93.30	12.90	151.07	93.30	9.56	111.90
	Probe32	87.00	12.94	153.26	87.44	9.59	113.53
CalTech	Probe41	96.15	14.21	55.70	96.15	11.12	43.58
	Probe42	97.52	14.54	57.35	97.52	11.38	44.87
	Probe43	98.78	14.64	57.85	98.78	11.46	45.27

TABLE 4: Performance of different probe sets for multiple enrolments of a subject in linear and kd-tree search.

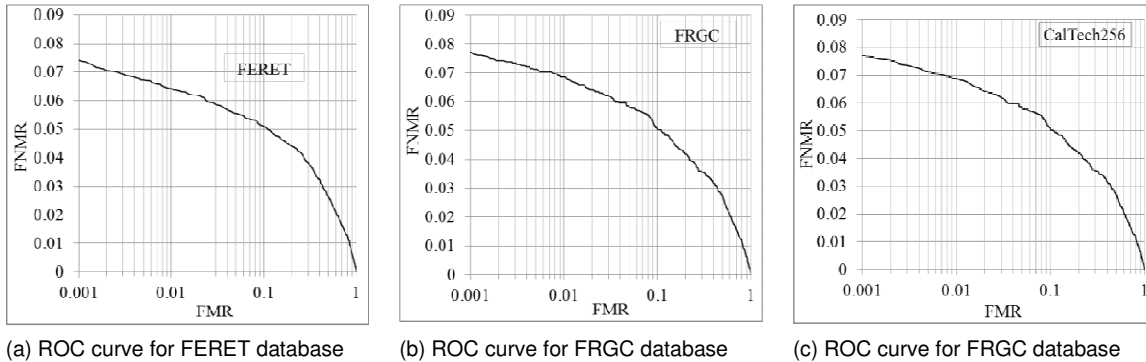


FIGURE 14: ROC curve for FERET, FRGC and CalTech256 databases.

5.5.6 Experiment 6: Searching time with different number of enrolled samples

In this section, we compute the searching time and the average number of comparisons required for linear and kd-tree based search techniques with the proposed index space. To compute these we enrolled different number of samples into the index space for FERET, FRGC and CalTech databases.

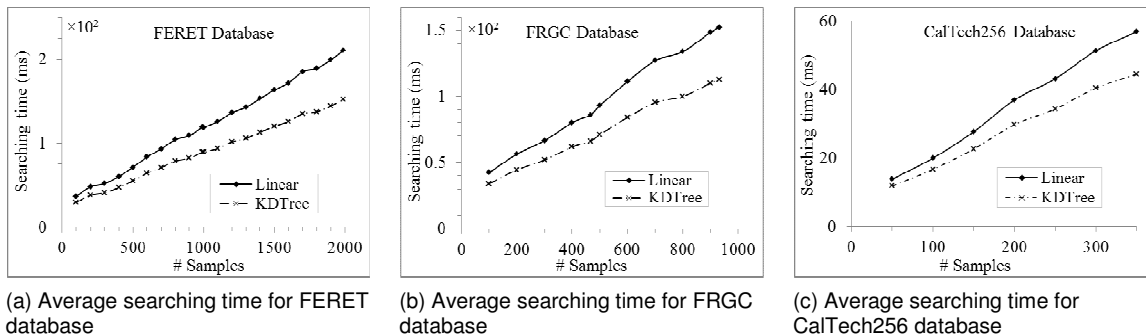
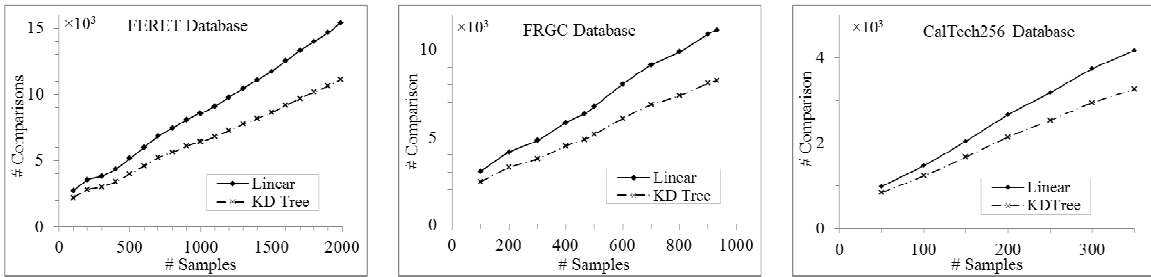


FIGURE 15: Average searching time with different sizes of databases for FERET, FRGC and CalTech256 databases.

The execution time (in Intel Core-2 Duo 2.00 GHz processor and 2GB RAM implementation environment) of linear and kd-tree search with FERET, FRGC and CalTech databases are shown in Fig. 15(a), (b) and (c), respectively. We observe that the execution time for kd-tree search is less than the linear search method. It is also observed that the rate of increment in execution time for kd-tree based search is less when the number of enrolled sample increases. We have given the average number of comparisons for linear and kd-tree based search in Fig. 16. From Fig. 16, we can see that the rate of increment in number comparisons is also less for kd-tree based search. Hence, we may conclude that to retrieve the similar identities for a given query, kd-tree based search within index cell is better than the linear search.

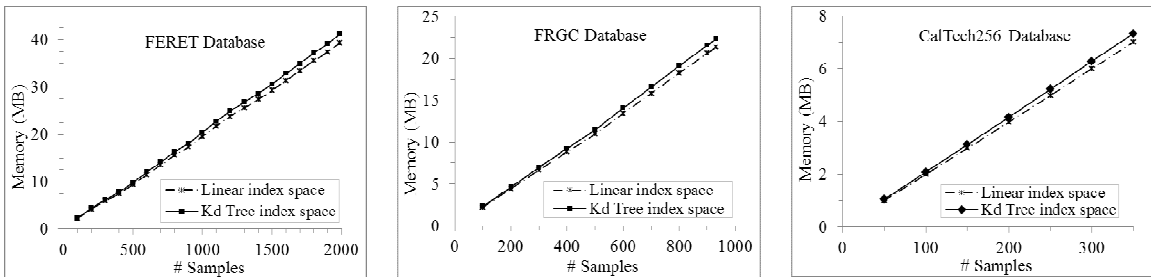


(a) Average number of comparisons for FERET database (b) Average number of comparisons for FRGC database (c) Average number of comparisons for CalTech256 database

FIGURE 16: Average number of comparisons with different sizes of databases for FERET, FRGC and CalTech256 databases.

5.5.7 Experiment 7: Memory for different number of enrolled samples

In our approach, 2 bytes are required to store the reference of index cube into a cell of first level index space and 4 bytes are required to store the reference of linear or kd-tree index space into a cell of index cube. There are 64 feature values in an index key and 4 bytes are required to store a feature value. We also store the identity of an individual with each index key. The identity field requires 2 bytes extra memory for each index key. We can store 216 identities with 2 byte identity field. Hence, a total of 258 bytes are required to store an index key along with the identity in linear index space. In kd-tree based index space, a single node of kd-tree requires 270 bytes memory. Fig. 17(a), (b) and (c) show the memory requirements for linear and kd-tree index spaces to store different number of samples for FERET, FRGC and CalTech databases, respectively. From Fig. 17 we observe that the memory requirements are almost same for the linear and kd-tree based index spaces.



(a) Memory requirements for FERET database (b) Memory requirements for FRGC database (c) Memory requirements for CalTech256 database

FIGURE 17: Memory requirements with different sizes of databases for FERET, FRGC and CalTech256 databases.

5.6 Discussion of Experimental Results

We have performed seven set of experiments to establish the accuracy and the efficiency of our proposed method. The first experiment is carried out to decide the parameter (number of the cells in an index cube). From this experiment, we observed that better penetration rate can be achieved by increasing the number of cells in index cube as it distributes the index key among more number of cells and it retrieved less number of keys at the time of querying. However, the probability of retrieving correct keys corresponding to the query is also reduced and it affects the hit rate of the identification system.

In experiment 2, we achieved less penetration rate when we apply indexing before identification. If we perform identification without indexing, query index keys are compared with the all stored

keys and the probability of matching is high at that time and we can achieve better hit rate but at the same time it increases the penetration rate. At the time of identification with indexing, probability of matching is reduced but the number of comparisons is also decreased which is reflected in the result of the second experiment.

There is an impact of sessions of capturing face images in the hit rate. From the third experiment, we can see that the performance of an indexing system is improved when we used the probe sets which are captured with less time gap. Further, the face image captured in indoor environment offers better performance. Further, if we consider the higher rank, the probability of matches will increase. The CMC curve shows the same trend of the hit rate in this experiment.

If we enroll multiple samples of a subject, the probability of matching a query subject will increase. From the fourth experiment, we observed that we achieved the maximum accuracy when we enrolled two samples per subject. It may be noted that accuracy of the system is improved if we enroll the samples with the different poses of a subject. Further, in the fifth experiment, we achieved better FMR and FNMR for the same reason.

As the searching complexity of the kd-tree is less than the linear search, we required less searching time when we used kd-tree in our indexing approach. The result of the sixth experiment substantiates that fact. If the number of samples for enrollment increases the memory requirement will also increase which we have shown in the seventh experiment.

5.7 Comparison with Existing Work

Lin et al. [28] propose an indexing structure to search the face from a large database. They compute a set of Eigenfaces based on the faces in the database. Then, they assign a rank to each face in the database according to its projection onto each of the Eigenface. Similarly, they compute the Eigenfaces for a query and rank a query face. Finally, they select a set of faces from the database corresponding to the nearest faces in the ranked position with respect to each Eigenface of the query face. These selected faces are used for recognition.

A linear subspace approximation method for face indexing has been developed by Mohanty et al. [29]. They build a linear model to create a subspace-based on the match scores. A linear transformation is applied to project face images into the linear subspace. For this purpose, first, they apply a rigid transformation obtained through principal component analysis and then a non-rigid affine transformation. An iterative stress minimization algorithm is used to obtain a distance matrix in a low-dimensional space and propose a linear out-of-sample projection scheme for test images. Any new face image is projected into this embedded space using an affine transformation.

Kaushik et al. [30] introduce a modified geometric hashing technique to index the face database. They extract features from a face image using SURF [27] operator. They apply mean centering, principal component analysis, rotation and normalization to preprocess the SURF features. Finally, they use geometric hashing to hash these features to index each facial image in the database.

We compare our approach with three existing face indexing approaches [28, 29, 30]. To compare our proposed work, we use *Gallery11*, *Gallery21* and *Gallery41* as gallery sets, and *Probe11*, *Probe21* and *Probe41* as probe sets for FERET, FRGC and CalTech databases, respectively. The comparison result is reported in Table 5. The comparison is done with respect to rank 1 hit rate, penetration rate and searching time. From Table 5 we can see that our approach gives better performance than existing approaches.

Approach	Performance	FERET	FRGC	CalTech
Lin et al. [28]	Hit Rate	83.87	85.19	73.08
	Penetration Rate	41.07	45.49	83.99
	Avg. Searching Time (ms)	2697.40	1738.35	895.27
Mohanty et al. [29]	Hit Rate	94.15	95.06	80.77
	Penetration Rate	25.58	24.95	60.5
	Avg. Searching Time (ms)	745.61	623.56	46.77
Kaushtik et al. [30]	Hit Rate	95.87	97.64	88.46
	Penetration Rate	16.96	18.61	27.49
	Avg. Searching Time (ms)	456.24	398.78	14.59
Proposed	Hit Rate	95.57	97.00	92.31
	Penetration Rate	7.90	12.55	23.72
	Avg. Searching Time (ms)	85.40	71.02	7.22

TABLE 5: Comparison of the proposed approach with existing approaches.

6. CONCLUSION AND FUTURE WORK

Face-based biometric identification system with a large pool of database requires huge computation time to search an individual's identity from the database. Best of our knowledge there is no good indexing technique exist for face identification system, which can identify a person in real-time when identification system is enrolled with a large number of users. In this work, we propose a new two-level indexing mechanism to reduce the search space for a face biometric-based identification system. We calculate a set of seventy dimensional index keys using SURF feature extraction method from a face image. Among seventy dimensions we consider only four dimensions to create the two-level index space. In the first level indexing, we group the index keys based on the sign of Laplacian value; and in the second level, we group the index keys based on the position and the orientation. We retrieve a set of similar identities for a query from the two-level index space using a hashing technique. The hashing technique requires $O(1)$ time complexity to retrieve the identities. We propose linear and kd-tree based searching mechanism to search the identities within the two-level index space. We have tested our approach with FERET, FRGC and CalTech face databases. The experimental result shows that kd-tree based search is performed better than the linear search. We can achieve 95.57%, 97% and 92.31% rank 1 hit rate with 7.90%, 12.55% and 23.72% penetration rate for FERET, FRGC and CalTech databases, respectively. Our approach gives better hit rate when multiple samples of a subject are enrolled into the database. We achieve on the average 8.21%, 11.87% and 24.17% search space reduction for different probe sets of FERET, FRGC and CalTech, respectively. With our proposed indexing approach, we achieve the computation time advantage without compromising the accuracy compared to traditional person identification systems.

The limitation of our approach is that it does not give good results under different poses (e.g. left or right profile) of face images. Our work can be extended to address the limitation said above. Further, in this work, we have targeted the face images captured in the indoor environment. This work can be utilized for the face images captured in outdoor environment.

7. REFERENCES

- [1] T. Huang, Z. Xiong and Z. Zhang. Handbook of Face Recognition. NY: Springer, 2011, pp. 617–638.
- [2] H. Lee, S.-H. Lee, T. Kim and H. Bahn. “Secure User Identification for Consumer Electronics Devices.” IEEE Trans. on Consumer Electronics, vol. 54, no. 4, pp. 1798–1802, 2008.
- [3] D.-S. Kim, S.-Y. Lee, B.-S. Kim, S.-C. Lee and D.-H. Chung. “On the Design of an Embedded Biometric Smart Card Reader.” IEEE Trans. on Consumer Electronics. vol. 54, no. 2, pp. 573–577, 2008.
- [4] “Physical access control biometrics.” Internet: <http://www.findbiometrics.com/physical-access/> [Jun. 12, 2012].
- [5] “Biometrics from Wikipedia.” Internet: http://en.wikipedia.org/wiki/Biometrics#Countries_applying_biometrics [Jul. 15, 2012].
- [6] W. Louis and K. N. Plataniotis. “Frontal Face Detection for Surveillance Purposes using Dual Local Binary Patterns Features,” in Proc. 17th IEEE International Conference on Image Processing (ICIP), Sept 2010, pp. 3809–3812.
- [7] P. Quintiliano and A. Rosa. “Face Recognition Applied to Computer Forensics.” The International Journal of Forensic Computer Science, vol. 1, no. 1, pp. 19–27, 2006.
- [8] A. K. Jain, B. Klare and U. Park. “Face Matching and Retrieval in Forensics Applications.” IEEE Multimedia, vol. 19, no. 1, pp. 20–28, 2012.
- [9] “Using the Iris Recognition Immigration System (IRIS).” Internet: <http://www.ukba.homeoffice.gov.uk/customs-travel/Enteringtheuk/usingiris/> [Jan. 20, 2012].
- [10] “Biometric Passport from Wikipedia.” Internet: http://en.wikipedia.org/wiki/Biometric_passport [Jun. 22, 2012].
- [11] “Unique Identification Authority of India.” Internet: <http://uidai.gov.in/> [Jul. 12, 2012].
- [12] “Identity Document from Wikipedia.” Internet: http://en.wikipedia.org/wiki/Identity_document#Bangladesh [Jul. 17, 2012].
- [13] Identity Card Policy, “List of National Identity Card Policies by Country.” Internet: http://en.wikipedia.org/wiki/List_of_identity_card_policies_by_country [Jun. 25, 2012].
- [14] “UIDAI Strategy Overview: Creating a Unique Identity Number for Every Resident in India.” Internet: http://uidai.gov.in/UID_PDF/Front_Page_Articles/Documents/Strategy_Overveiw-001.pdf [May 18, 2012].
- [15] A. Mhatre, S. Palla, S. Chikkerur and V. Govindaraju. “Efficient Search and Retrieval in Biometric Databases.” in Proc. SPIE Defense and Security Symposium, vol. 5779, Orlando FL, Mar. 2005, pp. 265–273.
- [16] M. Turk and A. Pentland. “Eigenfaces for Recognition.” Journal of Cognitive Neuroscence, vol. 3, no. 1, pp. 71–86, 1991.
- [17] A. Batur and M. Hayes. “Linear Subspace for Illumination Robust Face Recognition.” in Proc. IEEE International Conference on Computer Vision and Pattern Recognition, Dec 2001.
- [18] P. Yuen and J. Lai. “Face Representation Using Independent Component Analysis.” Pattern Recognition, vol. 35, no. 6, pp. 1247–1257, 2002.
- [19] P. Belhumeur, J. Hespanha and D. Kriegman. “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection.” IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 711–720, 1997.

- [20] J. Lu, K. Plataniotis and A. Venetsanopoulos. "Face Recognition Using LDA-Based Algorithms." IEEE Trans. on Neural Networks, vol. 14, no. 1, pp. 195–200, 2003.
- [21] N. Mittal and E. Walia. "Face Recognition Using Improved Fast PCA Algorithm." in Proc. Congress on Image and Signal Processing, Sanya, Hainan, 2008, pp. 554–558.
- [22] S. T. Roweis and L. K. Saul. "Nonlinear Dimensionality Reduction by Locally Linear Embedding." Science, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [23] T. Shakhunaga and K. Shigenari. "Decomposed Eigenface for Face Recognition under Various Lighting Conditions." in Proc. IEEE International Conference on Computer Vision and Pattern Recognition, Dec 2001.
- [24] W. Zhao, R. Chellappa, A. Rosenfeld and P. Phillips. "Face Recognition: A Literature Survey." ACM Computing Surveys, vol. 35, no. 4, pp. 399–458, 2003.
- [25] W. Yu, X. Teng and C. Liu. "Face Recognition Using Discriminant Locality Preserving Projections." Image and Vision Computing, vol. 24, no. 3, pp. 239–248, 2006.
- [26] X. He and P. Niyogi. "Locality Preserving Projections." in Proc. Conference Advances In Neural Information Processing Systems 16, 2003, pp. 153–160.
- [27] H. Bay, A. Ess, T. Tuytelaars and L. V. Gool. "SURF: Speeded Up Robust Features." Computer Vision and Image Understanding, vol. 110, no. 3, pp. 346–359, 2008.
- [28] K. H. Lin, K. M. Lam, X. Xie and W. C. Siu. "An Efficient Human Face Indexing Scheme Using Eigenfaces." in Proc. IEEE International Conference on Neural Networks and Signal Processing, 2003, pp. 920–923.
- [29] P. Mohanty, S. Sarkar, R. Kasturi and P. J. Phillips. "Subspace Approximation of Face Recognition Algorithms: An Empirical Study." IEEE Trans. on Information Forensics and Security, vol. 3, no. 4, pp. 734–748, 2008.
- [30] V. D. Kaushik, A. K. Gupta, U. Jayaraman and P. Gupta. "Modified Geometric Hashing for Face Database Indexing." in Proc. 7th international conference on Advanced Intelligent Computing Theories and Applications: with Aspects of Artificial Intelligence, 2011, pp. 608–613.
- [31] H. Wolfson and I. Rigoutsos. "Geometric Hashing: An Overview." IEEE Computational Science and Engineering, vol. 4, no. 4, pp. 10–21, 1997.
- [32] Y. Lamdan and H. J. Wolfson. "Geometric Hashing: A General and Efficient Model-based Recognition Scheme." in Proceedings of the 2nd International Conference on Computer Vision, 1988, pp. 238–249.
- [33] D. S. Bolme, J. R. Beveridge, M. L. Teixeira and B. Draper. "The CSU Face Identification Evaluation System: Its Purpose, Features and Structure." in Proc. 3rd International Conf. on Computer Vision Systems, 2003.
- [34] P. Viola and M. J. Jones. "Robust Real-time Face Detection." International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, 2004.
- [35] H. Bay, T. Tuytelaars and L. V. Gool. "SURF: Speeded Up Robust Features." in Proc. 9th European Conference on Computer Vision, vol. 3951, 2006, pp. 404–417.
- [36] T. Lindeberg. "Feature Detection with Automatic Scale Selection." International Journal of Computer Vision, vol. 30, no. 2, pp. 79–116, 1998.
- [37] A. Jensen and A. L. Cour-Harbo. Ripples in Mathematics: The Discrete Wavelet Transform. Springer, 2001.

- [38] J. L. Bentley. "Multidimensional Binary Search Trees Used for Associative Searching." *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
- [39] A. Moore. "A Tutorial on Kd-trees." Tech. Rep., University of Cambridge, 1991, Internet: <http://www.cs.cmu.edu/simawm/papers.html> [Nov. 20, 2011].
- [40] P. J. Phillips, H. Moon, S. A. Rizvi and P. J. Rauss. "The FERET Evaluation Methodology for Face Recognition Algorithms." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1090–1104, 2000.
- [41] "The Facial Recognition Technology (FERET) Database." Internet: [http://www.itl.nist.gov/iad/humanid/feret/feret master.html](http://www.itl.nist.gov/iad/humanid/feret/feret%20master.html) [Dec. 10, 2011].
- [42] "The Color FERET Database: Version 2." Internet: <http://www.nist.gov/itl/iad/ig/colorferet.cfm> [Feb. 10, 2012].
- [43] J. Phillips and P. J. Flynn. "Overview of the Face Recognition Grand Challenge." in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Jun, 2005.
- [44] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer and W. Worek. "Preliminary Face Recognition Grand Challenge Results." in *Proc. 7th International Conference on Automatic Face and Gesture Recognition*, 2006, pp. 15–24.
- [45] "Face Recognition Grand Challenge (FRGC)." Internet: <http://www.nist.gov/itl/iad/ig/frgc.cfm> [May 8, 2012].
- [46] G. Griffin, A. Holub and P. Perona. "Caltech-256 Object Category Dataset." Tech. Rep., California Institute of Technology, 2007.
- [47] "CalTech 256." Internet: http://www.vision.caltech.edu/Image_Datasets/Caltech256/ [Mar. 7, 2012]
- [48] "The Open Source Computer Vision (OpenCV) Library." Internet: <http://opencv.willowgarage.com/wiki/Welcome> [Dec. 16, 2011].
- [49] N. B. Puan and N. Sudha. "A Novel Iris Database Indexing Method Using the Iris Color." in *Proc. 3rd IEEE Conf. on Industrial Electronics and Applications (ICIEA 2008)*, Singapore, Jun 2008, pp. 1886–1891.
- [50] A. Gyaourova and A. Ross. "Index Codes for Multibiometric Pattern Retrieval." *IEEE Trans. on Information Forensics and Security*, vol. 7, no. 2, pp. 518–529, 2012.
- [51] B. Bhanu and X. Tan. "Fingerprint Indexing Based on Novel Features of Minutiae Triplet." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 616–622, 2003.
- [52] R. Cappelli, M. Ferrara and D. Maltoni. "Fingerprint Indexing Based on Minutia Cylinder-Code." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 1051–1057, 2011.
- [53] J. R. Beveridge, D. Bolme, B. A. Draper and M. Teixeira. "The CSU Face Identification Evaluation System." *Machine Vision and Applications*, vol. 16, no. 2, pp. 128–138, 2005.
- [54] S. Arya and D. M. Mount. "Algorithms for Fast Vector Quantization." in *Proc. of DCC '93: Data Compression Conference*, Snowbird, UT, 1993, pp. 381–390.
- [55] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman and A. Y. Wu. "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions." *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.

- [56] M. Muja and D. G. Lowe. "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration." in Proc. International Conference on Computer Vision Theory and Applications (VISAPP'09), 2009, pp. 331–340.
- [57] "FLANN - Fast Library for Approximate Nearest Neighbors." Internet: <http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN> [May 27, 2012].

INSTRUCTIONS TO CONTRIBUTORS

The *International Journal of Biometric and Bioinformatics (IJBB)* brings together both of these aspects of biology and creates a platform for exploration and progress of these, relatively new disciplines by facilitating the exchange of information in the fields of computational molecular biology and post-genome bioinformatics and the role of statistics and mathematics in the biological sciences. Bioinformatics and Biometrics are expected to have a substantial impact on the scientific, engineering and economic development of the world. Together they are a comprehensive application of mathematics, statistics, science and computer science with an aim to understand living systems.

We invite specialists, researchers and scientists from the fields of biology, computer science, mathematics, statistics, physics and such related sciences to share their understanding and contributions towards scientific applications that set scientific or policy objectives, motivate method development and demonstrate the operation of new methods in the fields of Biometrics and Bioinformatics.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJBB.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Started with Volume 7, 2013, IJBB appears with more focused issues related to biometrics and bioinformatics studies. Besides normal publications, IJBB intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

LIST OF TOPICS

The realm of International Journal of Biometrics and Bioinformatics (IJBB) extends, but not limited, to the following:

- Bio-grid
- Bioinformatic databases
- Biomedical image processing (registration)
- Biomedical modelling and computer simulation
- Computational intelligence
- Computational structural biology
- DNA assembly, clustering, and mapping
- Fuzzy logic
- Gene identification and annotation
- Hidden Markov models
- Molecular evolution and phylogeny
- Molecular sequence analysis
- Bio-ontology and data mining
- Biomedical image processing (fusion)
- Biomedical image processing (segmentation)
- Computational genomics
- Computational proteomics
- Data visualisation
- E-health
- Gene expression and microarrays
- Genetic algorithms
- High performance computing
- Molecular modelling and simulation
- Neural networks

CALL FOR PAPERS

Volume: 7 - Issue: 4

i. Paper Submission: November 30, 2013 **ii. Author Notification:** December 25, 2013

iii. Issue Publication: December 2013

CONTACT INFORMATION

Computer Science Journals Sdn Bhd

B-5-8 Plaza Mont Kiara, Mont Kiara
50480, Kuala Lumpur, MALAYSIA

Phone: 006 03 6207 1607
006 03 2782 6991

Fax: 006 03 6207 1697

Email: cscpress@cscjournals.org

CSC PUBLISHERS © 2013
COMPUTER SCIENCE JOURNALS SDN BHD
B-5-8 PLAZA MONT KIARA
MONT KIARA
50480, KUALA LUMPUR
MALAYSIA

PHONE: 006 03 6207 1607
006 03 2782 6991

FAX: 006 03 6207 1697

EMAIL: cscpress@cscjournals.org