5. Signature Recognition & Keystroke Dynamics

Signature verification is an important research area in the field of authentication of a person as well as documents in e-commerce and banking. We can generally distinguish between two different categories of signature verification systems: online, for which the signature signal is captured during the writing process, thus making the dynamic information available, and offline for which the signature is captured once the writing process is over and thus, only a static image is available.

In case of static systems morphological characteristics of signatures are used for forgery detection. In case of dynamic system the speed, pressure, acceleration based dynamic features make it possible to identify the signing style. This makes dynamic system more secure. Forgeries are possible in real life and they are difficult to identify in case of static systems.

We have implemented static signature recognition based on Vector Quantization [245],[249],[250] & cluster based features, morphological dilations based feature extraction [144], [247],[248]. The topic of the thesis is "Biometric Authentication Systems" & static signature recognition systems are mainly used for document verification rather than authentication. Hence we consider only dynamic (Online) signature recognition systems in this chapter.

5.1 Online Signature Data Capturing & Preprocessing

"Dynamic Signature" is a biometric modality that uses, for recognition purposes, the anatomic and behavioral characteristics that an individual exhibits when signing his or her name (or other phrase) [1], [3], [145], [146]. In On–line approach we can acquire more information about the signature which includes the dynamic properties of signature. We can extract information about the writing speed, pressure points, strokes, acceleration as well as the static characteristics of signatures [143]. This leads to better accuracy because the dynamic characteristics are very difficult to imitate, but the system requires user co-operation and complex hardware. Digitizer tablets or pressure sensitive pads are used to scan signature dynamically.

Dynamic Signature devices should not be confused with electronic signature capture systems that are used to capture a graphic image of the signature and are common in locations where merchants are capturing signatures for transaction authorizations. Data such as the dynamically captured direction, stroke, pressure, and shape of an individual's signature can enable handwriting to be a reliable indicator of an individual's identity (i.e., measurements of the captured data, when compared to those of matching samples, are a reliable biometric for writer identification.). We can also capture the static characteristics of signatures [143], [251], [252]. This leads to better accuracy because the dynamic characteristics are very difficult to imitate, but the system requires user co-operation and complex hardware [143]. Digitizer tablets or pressure sensitive pads are used to scan signature dynamically, one such tablet is shown in Fig.5.1.

(a) **(b)**

Fig. 5.1. Digitizer Tablet for Online Signature Scan (a) Wacom Intuos4 Digitizer Connected to PC (b) Scanning of Dynamic Signature

In the literature one can find a vast amount of work done for biometrics recognition as well as new directions are coming in sight for further research with development of faster machine and advanced sensors [1]. Development of such systems requires biometrics database, many biometrics databases are available on the internet for research purpose, like Fingerprint Database FVC 2000, 2002, 2004, 2006 Databases [202]. FERRET face database [253], CASIA Iris Database [254], Static Signatures Database [255]. Such databases have been widely used by researchers for the development of algorithms for biometrics recognition. But when

it comes for real time implementation we need to capture data for the biometrics sensors.

5.1.1 Capturing Data from Digitizer Device

We are using Wacom Intuos 4 Digitizer Tablet for capturing dynamic features of handwritten signature. Microsoft Visual C# 2005 (.NET Framework 2.0) is used for interfacing this device. Developing such interface application requires programming using Component Object Model (COM) as well as .NET assembly programming [191].

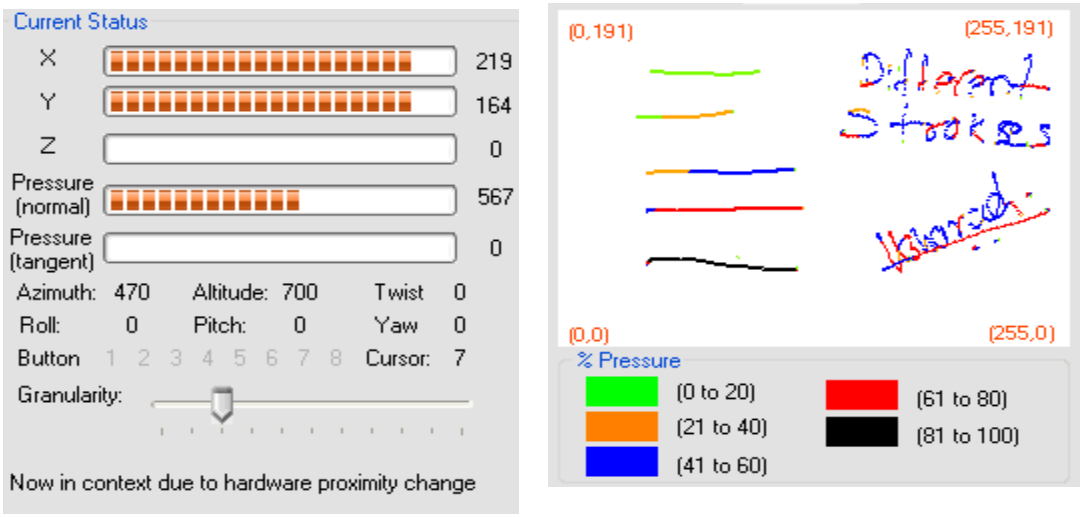
This Wacom Intuos 4 digitizer is mainly used by graphics designers but we discuss its use as a sensor for handwritten signatures. A specialty of this device is that along with conventional parameters this device also gives Z-co-ordinate of the tip of the pen while signing; this enables us to capture X,Y,Z co-ordinates of the signature in a 3 Dimensional space. Hence we say that it extracts multidimensional features of a signature. Note that the Z-Coordinate is sensed in a limit up to 2cm (approx) from the scanner surface; but it is sufficient to detect the co-ordinates of tip when user lifts the pen while signing. Typical features of the tablets are as follows

1. Active Area (W x H) 157.5 x 98.4 mm.
2. Connectivity-USB connectivity.
3. Pressure levels -2048.
4. Sensor pen without battery.
5. Minimum ON weight (Minimum weight sensed.
   by the pen tip) – 1Gram.
6. Report rate- 200 Points /Second.
7. LPI - lines per inch-5080 lpi.

The manufacturer has provided driver to install this device on to the operating system but we have designed the interface so that the above mentioned features are captured in our application for dynamic signature recognition. The data coming from device is coming in the form of data packets. We gate following features of the signature in captured data packet.

1. X,Y,Z Coordinates of the pen Tip.
2. Pressure – Pressure applied at the point
3. Tangent Pressure – tangent pressure of the tip.
4. Azimuth – Pen tip azimuth (corresponding to tip angle)

5. Altitude- Tip altitude corresponding to the different tip of the pen.
6. Packet Serial- Packet serial number
7. Packet Timing – Timestamp



(a) **(b)**

Fig. 5.2. (a) Captured Packed Data from Wacom Intuos 4 (b) Captured Pen Strokes & Signature

Some of the captured features are displayed in Fig. 5.2 (a) Captured pen strokes at different pressure levels are shown in Fig. 5.2 (b). Fig. 5.3 Shows some the plot for the data captured for signature given in Fig. 5.4 (c).

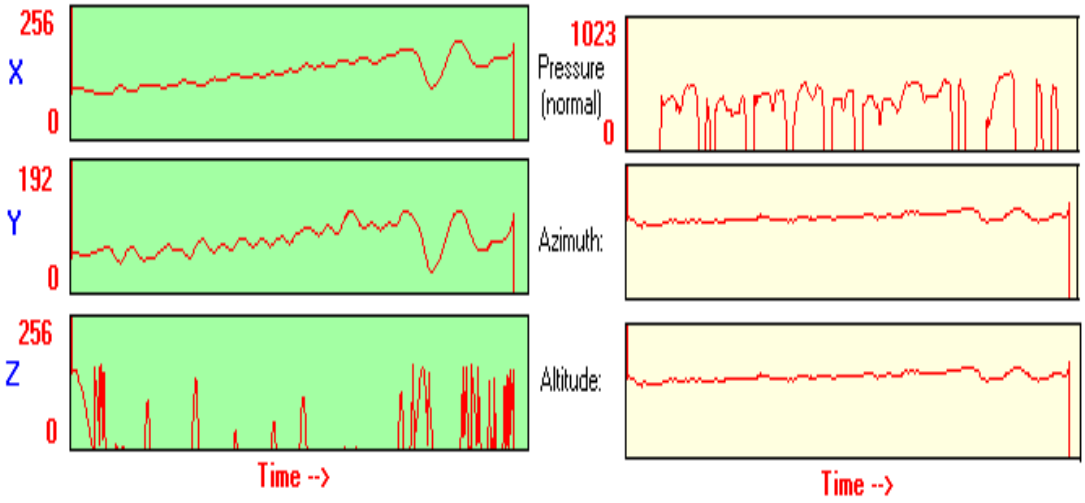


Fig. 5.3. Signature Feature Plot for Multidimensional features- X,Y,Z Co-ordinates, Pressure Azimuth & Altitude parameter

We have used this Digitizer to capture live X, Y, Z (3D) coordinates of the dynamic signature as multidimensional features along with conventional features like pressure, azimuth and 2D coordinates. Next step in the design of Dynamic SRS is

preprocessing the scanned signature data. In the next section we discuss this step.

5.1.2 Preprocessing Dynamic Signature Data

We have discussed various dynamic SRS approaches [145] – [155] in the Chapter 2 of Literature Survey. One thing that should be noted is that all these approaches need signature data with dynamic information. When the data comes from the hardware it is raw and we have to preprocess it to normalize the errors due to sampling, quantization, speed of hardware, signing position etc. We are using Wacom Intuos 4 for our experiments and we have also experienced the need of preprocessing the data. Doroz and Wrobel [259] have discussed this issue and proposed a technique of sampling the point uniformly to have equal number of points per unit time. They have used signature verification competition database [260].

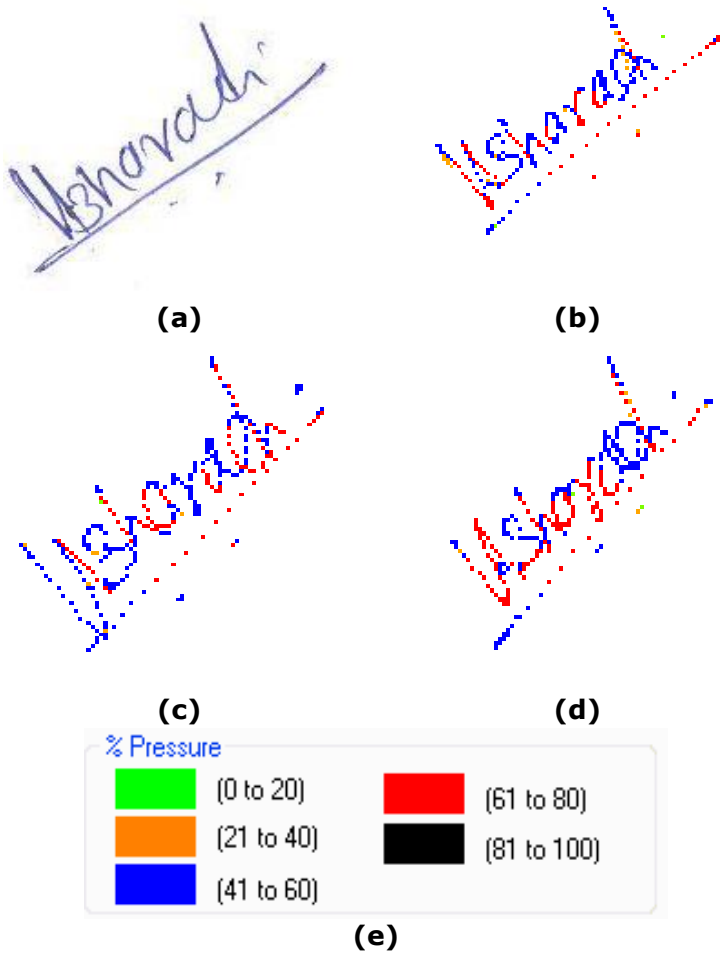


Fig. 5.4. Signature Samples of a Person (a) Static Scanned Signature, (b)(c),(d) Dynamic Signature Scanned by Wacom Intuos 4 (e) Pressure Levels for the Dynamic Signatures Shown.

As we are using hardware as discussed above to capture the signature, we have observed issues in captured signature. In the programmed interface we have observed that the interface could deliver only 100 Packets/Second. As the digitizer has finite rate of sampling and data transfer, it cannot capture all the points on a curve but captures finite points as per the sampling rate. This gives the results as shown in Fig. 5.4. There is loss of continuity in the captured points; we have a static scanned signature as shown in Fig, 5.4(a), same persons dynamic signatures are shown in Fig. 5.4(b), (c), (d); **different colours indicate different pressure levels** as shown in Fig.5.4 (e). We can clearly see the points that are sampled. If the signing speed is high then the captured points are less. One such situation is shown in Fig. 5.5. This causes loss of precision in the input data and may result in decreased accuracy of the matching algorithm.

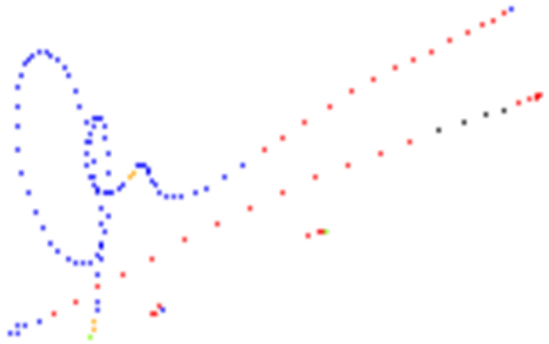


Fig. 5.5. Poorly Sampled Signature Due to High Signing Speed.

To solve this problem a method to calculate missing point's information and reduce the sampling error has been proposed by the authors. This method interpolates the captured points and calculates the missing information without loss of consistency. The method proposed is not same as normal interpolation as the missing points are on a curve and accuracy should be preserved in calculation as these points are on a biometric signature of a human being which can be used for authentication or authorization.

5.1.2.1 Proposed Technique for Preprocessing

Here we propose a technique that can be applied to any digitizer to calculate maximum possible points on the signature curve. Here we make use of the points in hand in order to calculate the parameters of points which are missing. Each captured points has multi-dimensional information. Each point contains information

about X, Y, Z-Coordinates, Pressure, Azimuth and Altitude of the pen tip. Hence i^{th} point P_i can be considered as:

$$Pi = Xi, Yi, Zi, Pri, Azi, Alti \quad (5.1)$$

We have to calculate the points between P_i & P_{i+1} if any points exist between them. To calculate the parameters we consider one fact that the points are consistent as they come from human behavior; and the consequent points tend to have values near to their neighbors' value. If we estimate X, Y & Z co-ordinates of missing points the remaining values can be taken as average between the P_i & P_{i+1} parameters. Proposed algorithm is divided into two steps.

1. Calculate X, Y Coordinates of missing points between P_i & P_{i+1} using Modified Digital Difference Analyzer (MDDA) Algorithm.
2. Estimate Z, Pressure, Azimuth, Altitude Values from the given interval points P_i & P_{i+1} by averaging.

We first discuss the proposed modified DDA Algorithm and then the averaging.

5.1.2.2 Modified Digital Difference Analyzer Algorithm (MDDA)

Digital Difference Analyzer Algorithm (DDA) is mainly used in computer graphics [261] for Line drawing on the raster. It calculates points on the line defined by two endpoints (X_1,Y_1) & (X_2,Y_2) .

It uses differential equation of a straight line.

$$\frac{dy}{dx} = m \quad (5.2)$$

$$\frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} \quad (5.3)$$

The intermediate points can be calculated as

$$x_{i+1} = x_i + \Delta x \quad (5.4)$$

$$y_{i+1} = y_i + \frac{y_2 - y_1}{x_2 - x_1}\Delta x \quad (5.5)$$

In the literature we can see that this algorithm is widely used for the scan conversion of a line. In our application we have different domain, we use this algorithm for finding missing points between the two interval points, these points are actually on a curve but as the distance between two points is very small (generally 0 to 4

Pixels maximum), hence the points can be assumed on a line and DDA can be used for calculation of points which lie one a line between two given points P_i & P_{i+1}. **We have made the following changes in the algorithm**.

1. In case of $X_2 < X_1$ of the two points (X_1,Y_1) & (X_2,Y_2) , the two points are interchanged for calculation, we use a flag here to mark the condition as we do not want interchange of the points as they lie on a signature curve.
2. In DDA the points are calculated and plotted on the screen, we use one two-dimensional array to store (X,Y) co-ordinates calculated.
3. If at the start of calculations the two points are interchanged, we check the flag and reverse the array content order to have actual sequence of the point on the signature.

The proposed MDDA Algorithm as follows :

1. Read the interval points (x_1,y_1) & (x_2,y_2)
if (x1 > x2) then exchange the points;
Set the flag reverse = true.

2. Calculate the difference.
dx = x2 - x1;
dy = y2 - y1;

3. If Abs(dx) > Abs(dy) then
m = dy / dx;
y = y1;
Calculate the points between X1 & X2 by
for (x = x1; x < x2; x++)
{
Xi = Round(x);
Yi = Round(y);
Store Xi,Yi to array Points[,];
y = y + m;
}
else
Go to Step 4;
Go to Step 6.

4. If Abs(dx) < Abs(dy) then
m = dx / dy;
x = x1;
if (y1 < y2) then
Calculate the points between Y1 & Y2 by

```
        for (y = y1; y < y2; y++)
        {
          Xi = Round(x);
          Yi = Round(y);
          Store Xi,Yi to array Points[,];
           x = x + m;
         }

    else
        Go to Step 5;
```
Go to Step 6.

5. If (y1 > y2) then
Calculate the points between Y1 & Y2 by

```
for (y = y1; y > y2; y--)
 {
   Xi = Round(x);
   Yi = Round(y);
   Store Xi,Yi to array Points[,];
   x = x - m;
  }
```

6. If reverse flag is set then
Rearrange the Points[,] array in reverse order.

7. Insert the points (Xi,Yi) from Points[,] array into the main signature features array between P_i & P_{i+1}.

In the next section we discuss the calculation of other parameters.

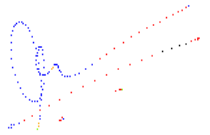
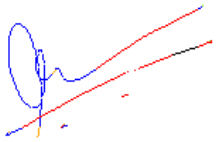
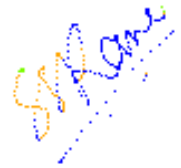
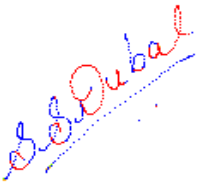
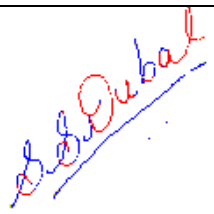
5.1.2.3 Calculating Z-coordinate, Pressure, Azimuth and Altitude

In the previous section we have discussed Modified DDA algorithm to calculate X & Y co-ordinates of points between two signature points P_i & P_{i+1}, still we have to calculate the other parameters of the interpolated points. We have to calculate Z Co-ordinate, Pressure, Azimuth, and Altitude of the points given by previous step. The signature points have temporal locality; means the consecutive points tend to have similar value as their neighbors. The maximum packet rate is 200 packets/seconds; it has been observed that the time difference between two sampled points varies from 5ms to 10ms. Hence we use this fact to interpolate the other parameters. We calculate the Z-coordinate (Distance of pen

tip from the digitizer surface), Pressure, Azimuth, Altitude (Azimuth & Altitude give information about pen tip angle whine signing).

Table 5.1

Interpolation Results for Different Signatures with Their Parameters and Calculation Time in milliseconds

Sampled & Interpolated Signature	Packet Count	Center of Mass of Signature (Cx,Cy)	Slope Angle	Total Pixel Count	Signature Arch Length in Pixels	Calculation Timing ms / Signature Timing ms
 Sampled Signature 1	157	103, 95	12.8	132	842	31.25
 Interpolated Signature 1		122, 100	11.18	246	907	1140
 Sampled Signature 2	227	132, 96	30.69	255	637	15.625
 Interpolated Signature 2		119, 92	29.38	349	649	1500
 Sampled Signature 3	709	123, 96	38.66	473	995	46.875
 Interpolated Signature 3		123, 96	38.02	530	999	2500

We calculate the values by taking Average of the parameters of P_i & P_{i+1}. To find the parameters of n^{th} (P_n) point between P_i & P_{i+1}, we use following equations.

$$z_n = \frac{z_i + z_{i+1}}{2} \tag{5.6}$$

$$\mathrm{Pr}_n = \frac{\mathrm{Pr}_i + \mathrm{Pr}_{i+1}}{2} \tag{5.7}$$

$$Az_n = \frac{Az_i + Az_{i+1}}{2} \tag{5.8}$$

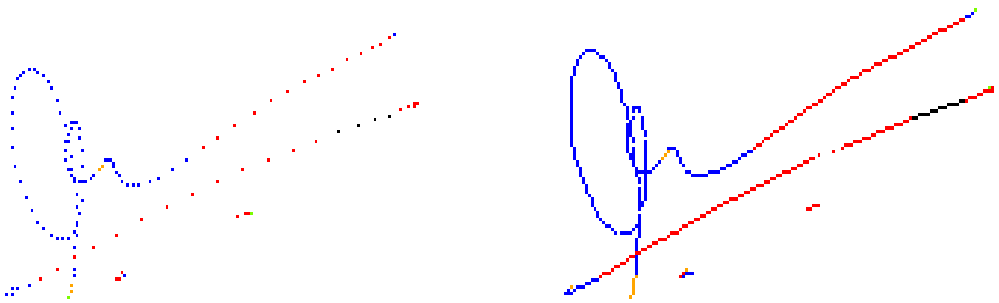
$$Alt_n = \frac{Alt_i + Alt_{i+1}}{2} \tag{5.9}$$

Thus we get complete set of information between the sampled points P_i & P_{i+1}. Each point has the format as follows,

$$Pi = Xi, Yi, Zi, Pri, Azi, Alti \tag{5.10}$$

5.1.2.4 Results for Preprocessing of Dynamic Signature

In Fig. 5.6 (a) we can see the captured signature which has errors due to sampling of points, we calculate the missing points using proposed scheme and the interpolated set of points give the signature as shown in Fig 5.6(b), we can clearly see that the Signature is fairly continuous.



(a) **(b)**

Fig. 5.6. Results of MDDA based Interpolation of Dynamic Signature (a) Captured Signature (b) Interpolated Signature

The proposed algorithm was tested on 30 different signatures at different speeds. We show some of the signatures and their interpolated form in Table 5.1. We can see that because of interpolation the pixel count has drastic change as the missing pixels are calculated but the global features like the Signature Arc length; Slope Angle and the center of mass have less change in

sampled and interpolated version. The interpolated signatures are continuous and can be further used for testing any dynamic signature recognition algorithm. Signature 1 is done at faster rate, signature 2 is done at moderate rate and signature 3 is done at slower rate. Signature 1 has maximum loss of continuity and has more points being calculated by MDDA based preprocessing, while signature 3 requires less points to be interpolated.

In Fig. 5.7 we plot the graph of signature time (signing time of signature) and the time required for the calculation of missing points; i.e. interpolation time. The interpolation time is very less and was observed between 10ms to 50ms. This makes the algorithm very attractive for real time implementation. The program was tested on AMD Athlon 64 Processor running at 1.8GHz, 1.5 GB RAM and Windows XP SP3 Operating System. Application programming platform is Visual C# 2005, .NET Framework version 2.0.

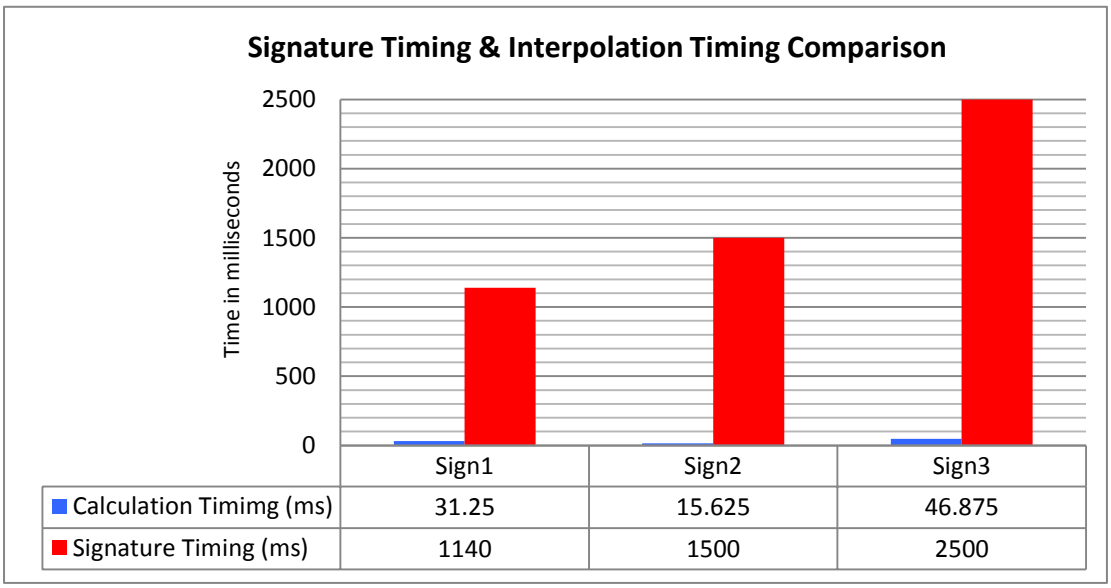


Fig. 5.7. Signature Time & Interpolation Timing Comparison (X-Axis- Signature Sample, Y Axis- Calculation Time in milliseconds)

In this section a novel dynamic signature preprocessing algorithm is discussed. This algorithm is based on proposed Modified Digital Difference Analyzer (MDDA) technique. The proposed algorithm is hardware independent and can be used with any type of digitizer tablet. This algorithm gives missing point because of finite data transfer & sampling rate of the hardware. The algorithm is fast and required maximum 50 milliseconds under testing, hence it is very attractive for real time use. The signature templates generated are continuous and any existing Dynamic

Signature recognition technique can use this method as their preprocessing step for better accuracy. In the next section we discussed online signature recognition system using the proposed hardware interface & preprocessing technique.

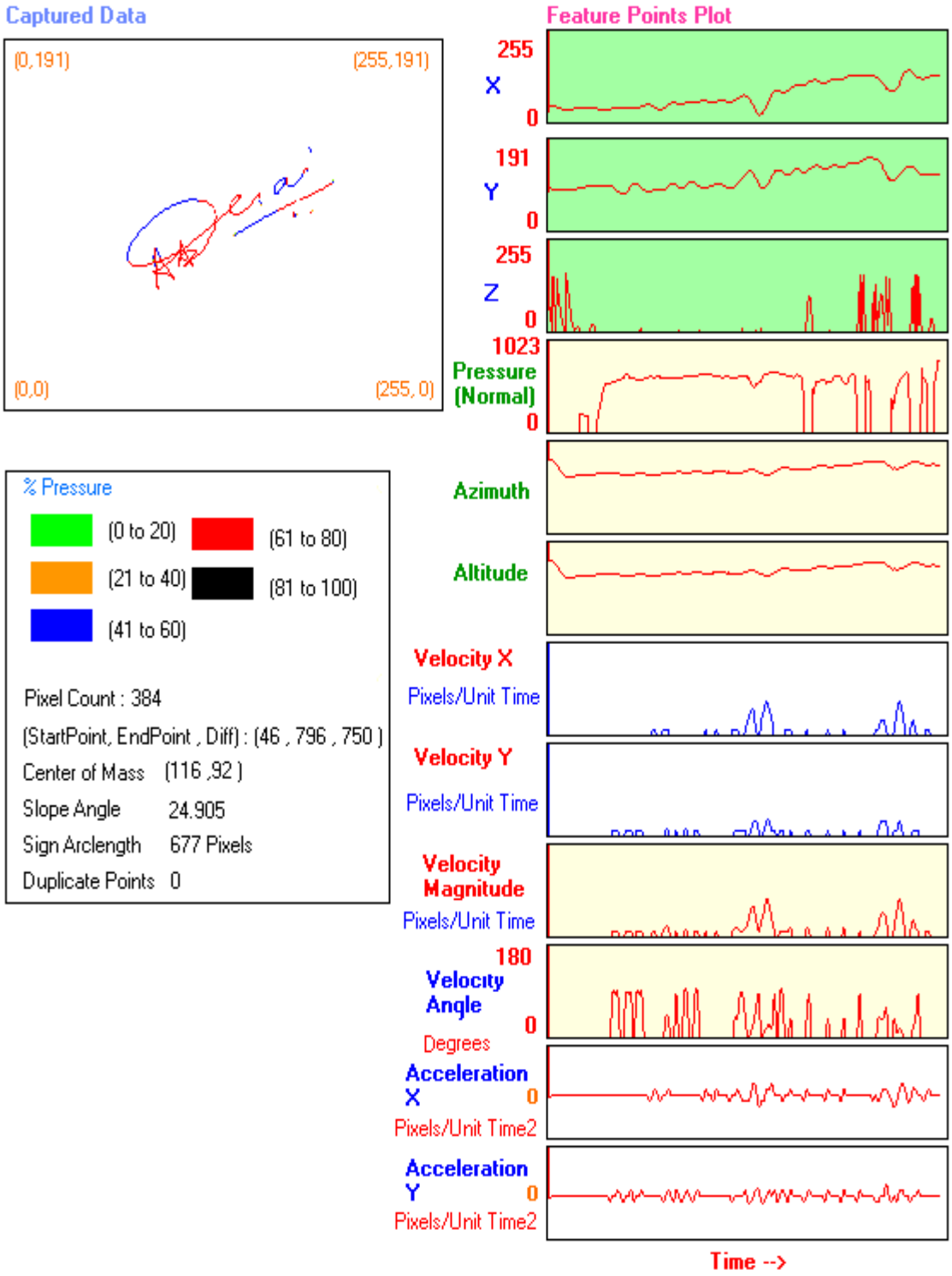


Fig. 5.8. A Typical Captured & Preprocessed Dynamic Signature, Its Pressure Map, Signature Parameters & Corresponding Feature Plot

5.1.3 Feature Points of Dynamic Signature

We have discussed data capture & preprocessing steps of online signature recognition systems. We have to perform feature extraction on this captured signature points. The captured & interpolated signature points have the form of $Pn = Xn, Yn, Zn, Prn, Azn, Altn$ (X, Y, Z Co-ordinates, Pressure, Azimuth, and Altitude). We extract some extra feature from the signature points these feature are shown in Fig. 5.8.

We have the basic features as X, Y & Z-Coordinates, Pressure, Azimuth & Altitude. We derive following features from this data.

1. Velocity X (V_x) & Velocity Y (V_y)- This is the velocity of the pen tip in x direction. Given by,

$$V_x = \frac{dx}{dt} \tag{5.11}$$

Discrete velocity points (V_{xn}) are calculated by dividing the signature points into equal intervals (Δt) we consider interval of 4 sample points.

$$Vx_n = \frac{X_{n+1} - X_n}{\Delta t} \tag{5.12}$$

Similarly V_y can be calculated.

2. Velocity Magnitude (V_{Mn}) – This Parameter gives the vector addition of V_x and V_y velocity vectors.

$$V_{Mn} = \sqrt{(V_{xn}^2 + V_{yn}^2)} \tag{5.13}$$

3. Velocity Angle ($V_{\theta n}$) – This is the instantaneous direction of velocity. Given by,

$$V_{\theta n} = \tan^{-1}\left(\frac{V_{yn}}{V_{xn}}\right) \tag{5.14}$$

4. Acceleration X (A_x) & Acceleration Y (A_y) - This is the acceleration of the pen tip in x direction. Given by,

$$A_x = \frac{d^2x}{dt^2} \tag{5.15}$$

Discrete acceleration points (A_{xn}) are calculated by dividing the velocity points into equal intervals (Δt) we consider interval of 4 sample points.

$$Ax_n = \frac{Vx_{n+1} - Vx_n}{\Delta t} \tag{5.16}$$

Similarly A_{yn} can be calculated.

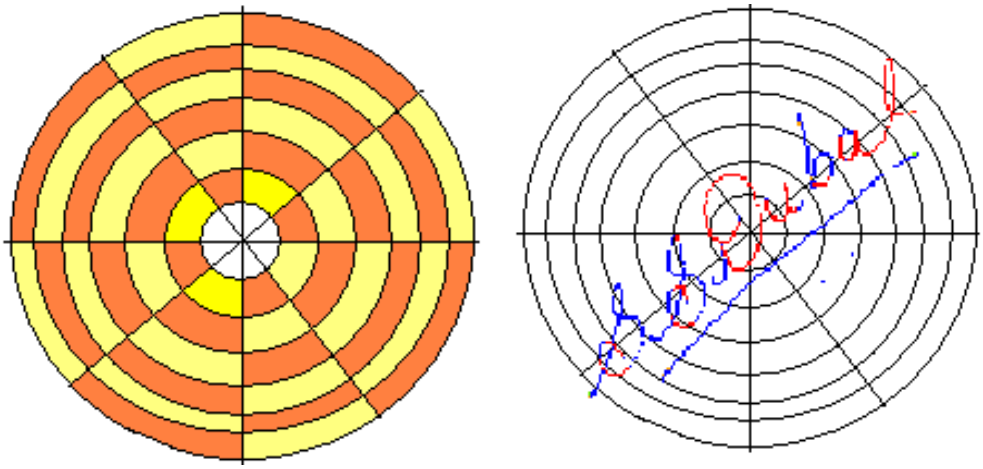
We use this feature points for generating the feature vector to match the dynamic signature. Summarizing we have total 12 parameters as X,Y & Z-Coordinates, Pressure (P_n), Azimuth (Az_n), Altitude(Al_n), Velocity-X (Vx_n), Velocity-Y (Vy_n), Velocity Magnitude (VM_n), Velocity Angle($V\theta_n$), Acceleration X (Ax_n) & Acceleration Y (Ay_n). Besides we also have Signature Pixel Count (PC), Arch Length (ARCL) & Slope Angle (S_θ) as Global features. Next we discuss online signature recognition using Gabor filter based feature extraction on the above mentioned feature points.

5.2 Online Signature Recognition Using Gabor Filter Based Features

We have previously used Gabor Filters in Section 3.1.1.2 for Fingerprint Segmentation and in Section 4.1.1 for face feature vector extraction. In this section we are implementing this tuned Gabor filter on the signature pressure map to generate the feature vector. For each signature pressure map image of size W × H centered at (*X*,*Y*), with W & H even, we extract the Gabor Magnitude [207] as follows for k = 1, . . . ,m:

$$g(X,Y,\theta_k,f,\sigma_x,\sigma_y) = \left| \sum_{x_0=-W/2}^{(W/2)-1} \sum_{y_0=-W/2}^{(W/2)-1} I(X+x_0, Y+y_0) h(x_0, y_0, \theta_k, f, \sigma_x, \sigma_y) \right| \tag{5.17}$$

Where, I(x, y) denotes the normalized pressure level of the signature pixel (x, y). As a result, we obtain *m* Gabor features for each signature pressure map. Only change is that instead of W x W image block we apply the Gabor filter to full signature pressure map template. We consider both the signature shape and pressure information in eight directions to calculate the feature vector. We use the technique in [36] on our pressure map of the signature and later incorporate the timing information with it. We use the tessellation with center aligning with the center of mass of the signature. The signature area is divided into total 48 sectors as shown in Fig. 5.9.



(a) (b)

Fig. 5.9. (a) 48 Sectors Tessellation Map Orientation (b) Tessellation Put Over the Signature for Gabor Filter Based Feature Extraction (This Fig. is for Representation Purpose Only)

Let $I(x, y)$ denote the pressure level at pixel (x, y) in an M X N signature image and let (Xc, Yc) denote the center point. The spatial tessellation of the signature space which consists of the region of interest is defined by a collection of sectors S_i, where the i^{th} sector S_i is computed in terms of parameters (r, q) [36] as follows,

$S_i = \{(x, y) \mid b\,(T_i + 1).\, r < b\,(T_i + 2),\ \theta_i <= \theta <= \theta_{i+1},$
$1 <= x <= N,\ 1 <= y <= M\}$ (5.18)

Where (b= 10 Pixels, K=8 Sectors in each band) and

$$T_i = i \text{ div } k \tag{5.19}$$

$$\theta_i = (i \mod k)\left(\frac{2\pi}{k}\right) \tag{5.20}$$

$$r = \sqrt{x - x_c}^{\,2} + \sqrt{y - y_c}^{\,2} \tag{5.21}$$

$$\theta = \tan^{-1}\left(\frac{y - y_c}{x - x_c}\right) \tag{5.22}$$

When we apply Gabor filter on the Signature pressure map, we have total 8 Planes of Gabor response corresponding to 8 Angles (K=8). Now using the above mentioned tessellation we calculate the mean and standard deviation of the Gabor response of the pixels of each sector (S_0 to S_{47}), and then we calculate standard deviation of Gabor response for each sector. In this way we get all 48 values of Gabor SD for one plane, such 8 planes are there hence the feature vector contains 48x8 Elements.

5.2.1 Adding Timing Information to the Tessellation Map

To add dynamic property to the Gabor Tessellation, we generate the timing information about each sector. We have time stamp for each pixel when captured. In each sector we find pixel with lowest time stamp and assign that timestamp to the sector. If any sector does not have pixels then the time stamp holds infinite value (∞) indicating omission of the sector.

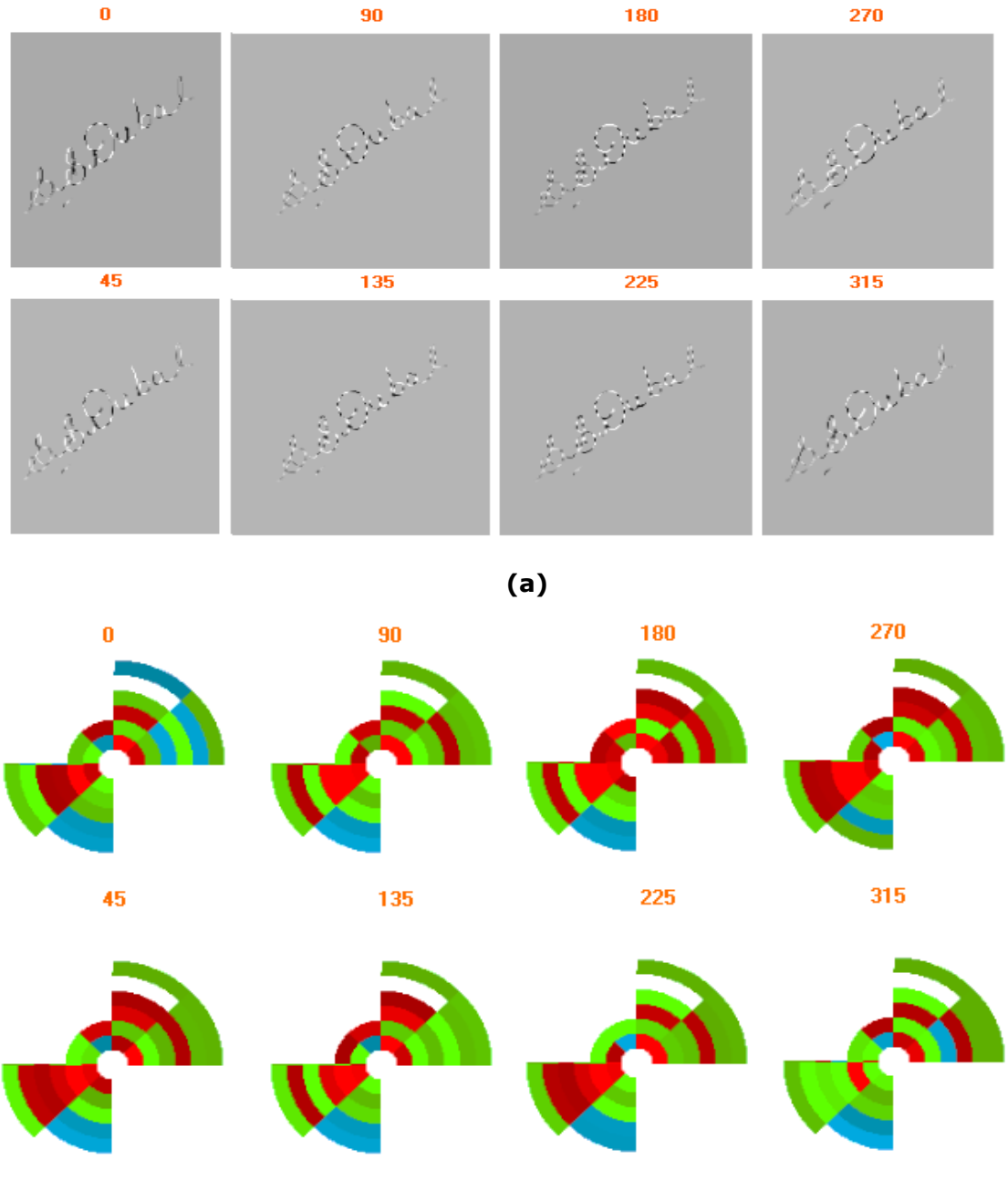


Fig. 5.10. (a) Gabor Response of a Signature for 8 Angles as shown on Top of Each Image (b) Corresponding Gabor Feature Vector

Blue color Indicates Low Values, Green and Red Indicate Increasing Standard Deviation of Gabor Filter Response

While evaluating Euclidian distance the sector values are sorted as per the timestamp and distance is calculated. As we have time sequenced vector we have also used a method called Dynamic Time Warping (DTW) for evaluating the similarity. For the signature shown in Fig. 5.9 the Gabor Response and corresponding Feature vector is shown in Fig. 5.10. Gabor response along 8 different angles in the range of [0-2Л] is shown in Fig. 5.10 (a), we extract standard deviation of the Gabor response of the pressure map tessellation for each sector. The corresponding Gabor feature map is shown in Fig. 5.10 (b). The feature vector is generated form these values along with the timing information for each sector. As discussed earlier we have total 48x8=384 values in the feature vector. Plot of typical feature vector is displayed in Fig. 5.11. This will be used for matching the signature, this feature vector contains both the texture as well as pressure information of the signature and timing information can be added by considering the timestamp of each sector as discussed above.

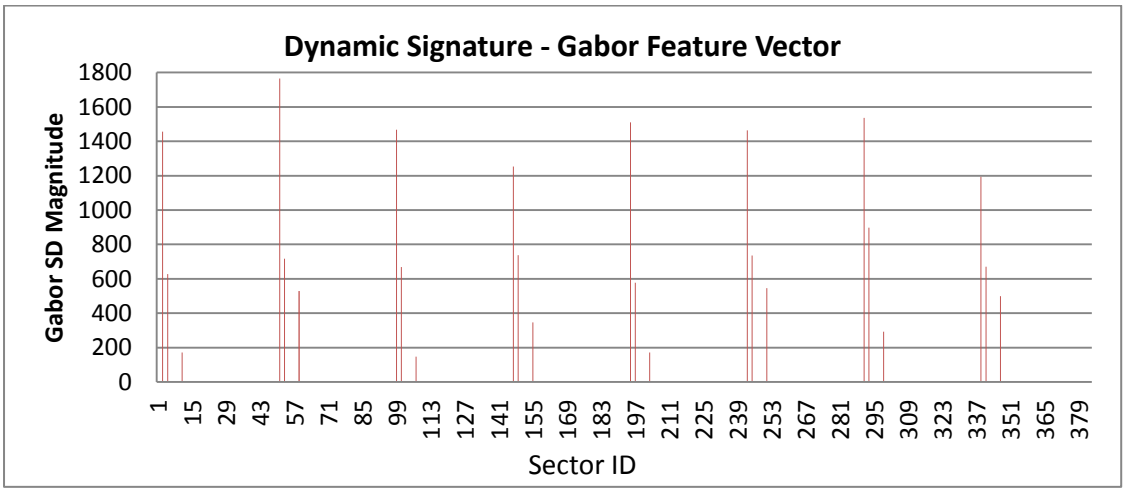


Fig. 5.11. Typical Signature Feature Vector Plot based on Gabor Response of Pressure Map Tessellation

5.2.2 Dynamic Time Warping (DTW)[262]

For evaluating two feature vectors with time stamp we have used Dynamic Time Warping (DTW) along with the Euclidian Distance (ED). Dynamic time warping is an algorithm for measuring similarity between two sequences which may vary in time or speed. For instance, similarities in walking patterns would be detected, even if in one video the person was walking slowly and if in another he or she were walking more quickly, or even if there were accelerations and decelerations during the course of one observation. DTW has been applied to video, audio, and graphics indeed, any data which

can be turned into a linear representation can be analyzed with DTW. In general, DTW is a method that allows a computer to find an optimal match between two given sequences (e.g. time series) with certain restrictions. The sequences are "warped" non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear variations in the time dimension. This sequence alignment method is often used in the context of hidden Markov models.

One example of the restrictions imposed on the matching of the sequences is on the monotonicity of the mapping in the time dimension. Continuity is less important in DTW than in other pattern matching algorithms; DTW is an algorithm particularly suited to matching sequences with missing information, provided there are long enough segments for matching to occur. We use DTW to find similarity between two Dynamic Signature Feature vectors (SFV) with timestamp.

5.2.3 Results for Gabor & DTW

The signatures are verified by evaluating the Euclidian distance between the feature vectors of signature templates. We have implemented this method in MS Visual C# 2005 (.NET Framework 2.0) on an AMD Athlon FX 64 processor at 1.8 GHz and 1.5 GB RAM running Windows XP SP3.

For testing we have collected 250 signatures from 25 different users in average two sittings with time difference of one to two weeks. Volunteers were asked to forge Signatures. The result analysis is presented here. We have tested three options

A. Gabor Filter based feature vector without Timestamp.

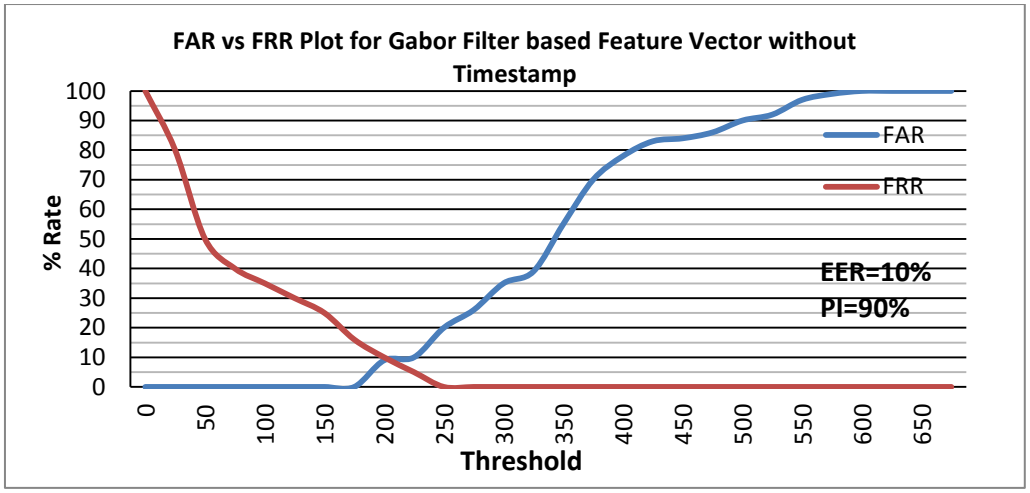
B. Gabor Filter based feature vector with Timestamp & ED.

C. Gabor Filter based feature vector with Timestamp & DTW.

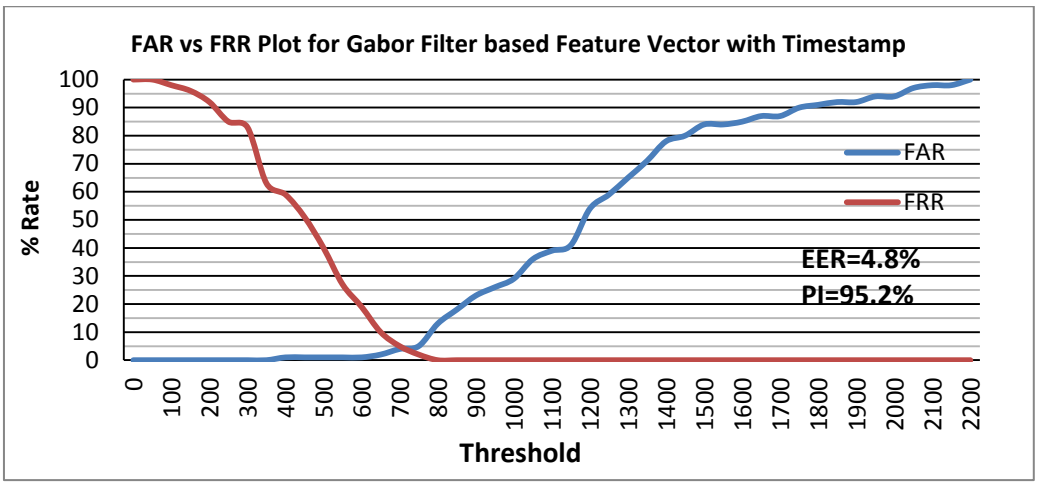
We have used Euclidian Distance (ED) as well as Dynamic Time Warping (DTW) between two vectors. Intra class as well as inter class testing is performed on the signature database. Total 493 tests were performed for intra-class matching & 1025 tests were performed for inter-class matching. The results are summarized in Fig. 5.12.

A. Gabor Filter based Feature Vector without Timestamp

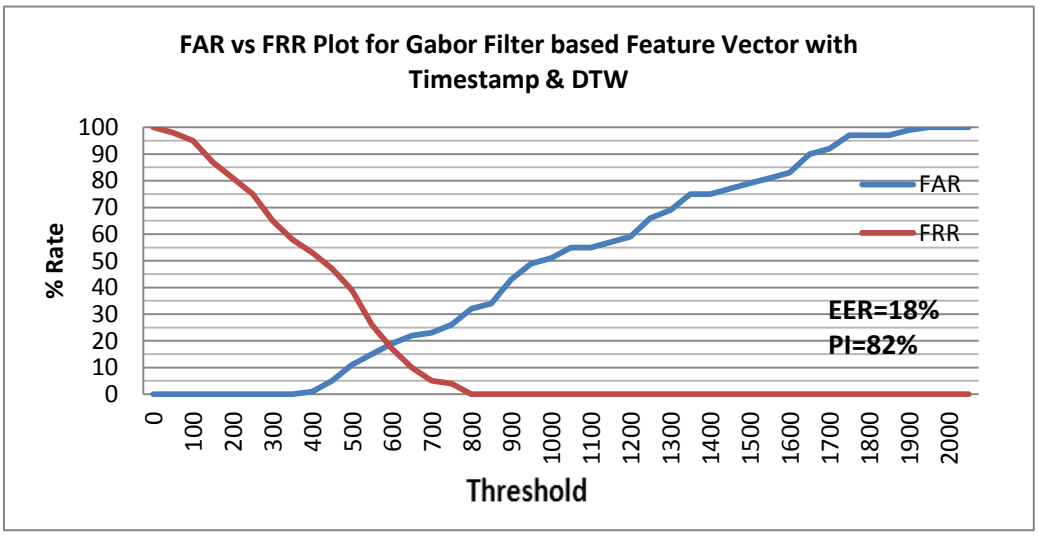
The analysis shows that the Performance Index (PI) for Gabor Filter based Feature vector based classification is 90 %. EER is 10 % for as shown in Fig 5.12(a) & (b).



(a)



(b)



(c)

Fig. 5.12. Result Analysis for Signature Recognition using Gabor Filter (FAR-FRR Plots) (a) Gabor Filter based Feature Vector without Timestamp (b) Gabor Filter based Feature Vector with Timestamp (c) Gabor Filter based Feature Vector with Timestamp & DTW

B. Gabor Filter based Feature Vector with Timestamp & ED

We have performed same analysis of the signature verification using Gabor filter based feature vector with timestamp. The performance improved as the feature vectors distance is evaluated by adding the timing information of the signature. The analysis shows PI is 95.2% and the EER is 4.8%.

C. Gabor Filter based Feature Vector without Timestamp & DTW

Here we have evaluated DTW distance between two Signature Feature Vectors. It was found that the DTW is not giving improvement but performance degrades. The analysis shows that PI is 82% and the EER is 18%.

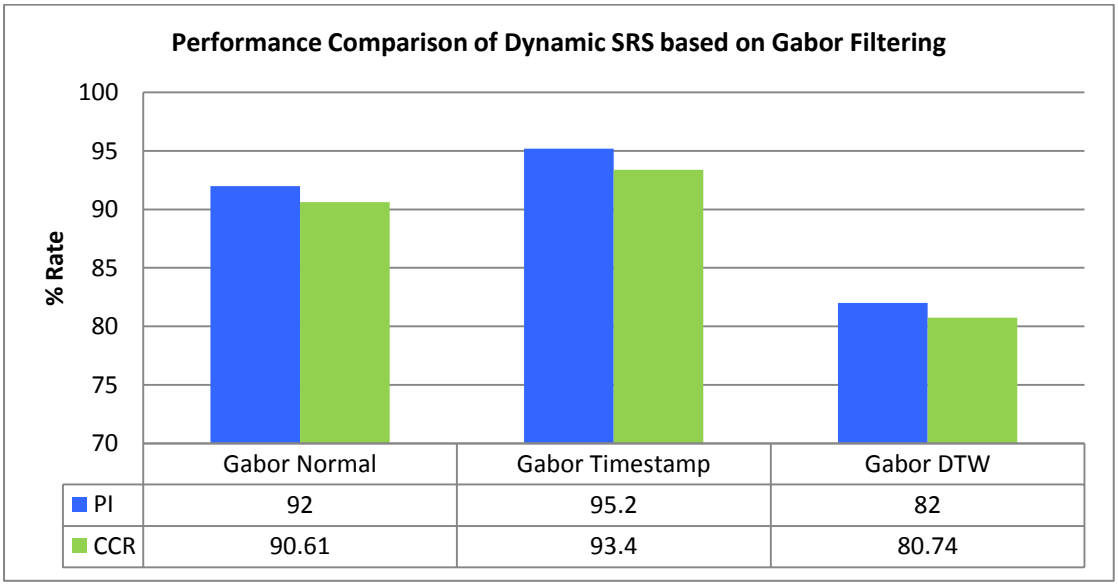


Fig. 5.13. Performance Comparison of The Gabor Filter based Online Signature Recognition Systems

Finally we can say that the signature feature vector with pressure & timing information gives the best result with Euclidian Distance (ED) as distance measure. The summary is given in Fig. 5.13. In this section we have discussed a feature vector based on Gabor filtering for the dynamic signature recognition. The algorithm is tested on live signature capturing on Wacom Intuos 4 Digitizer Tablet. The timing information was also incorporated into the Gabor filter based feature vector by Dynamic Time Warping (DTW). The method gives 90% PI & 90.61% CCR for Gabor feature vector without timestamp. When timing information is added the performance is improved and the reported PI is 95.2% & CCR is 93.4%. DTW is not giving any performance improvement. The performance can be further improved by use of good classifier. In the next section we will discuss online signature recognition using

clustering algorithm. We have previously used KFCG & KMCG for feature extraction of face & iris biometrics we extend this technology for the dynamic signature template as captured from the digitizer.

5.3 Online Signature Recognition Using VQ

Feature Extraction is main step is signature recognition. We are using vector quantization based scheme for feature vector generation. This scheme is fast and giving good results. As the dynamic signature recognition mainly deals with time based behavior of signature we are using Vector Quantization along the time axis of the signature, the captured data is in following form, each point contains information about X, Y, Z-Coordinates, Pressure, Azimuth and Altitude of the pen tip. Hence i^{th} point P_i can be considered as (Ref. Eqn. 5.23)

$$P_i=\{X_i,Y_i,Z_i,T_i,Pr_i,Az_i,Alt_i\} \quad (5.23)$$

'T_i' is the timestamp or sequence number of specific point in the signature. This is a multidimensional feature vector, we implement clustering across the Time axis (Timestamp) of the signature using Kekre's Median Codebook Generation Algorithm (KMCG) & Kekre's Fast Codebook Generation Algorithm (KFCG). We are using these techniques mainly because they have given best performance in case of Face & Iris Recognition. In section 5.1.3 we have discussed many parameters for dynamic signature as X,Y & Z-Coordinates, Pressure (P_n), Azimuth (Az_n), Altitude(Al_n), Velocity-X (Vx_n), Velocity-Y (Vy_n), Velocity Magnitude (VM_n), Velocity Angle($V\theta_n$), Acceleration X (Ax_n) & Acceleration Y (Ay_n). But for clustering only six parameters along with timestamp are considered. Because the velocity, velocity angle & acceleration based parameters are calculated for time intervals (dt=4) and not for all points, these parameters will be considered in next section.

We have previously discussed a method of preprocessing online signature template to compensate sampling error by interpolation. We analyze the effectiveness of this method by applying the clustering on both the original and interpolated template.

5.3.1 Feature Vector Generation by KFCG & KMCG

These algorithms reduce the time for code book generation [247] which will be used as feature vector. KMCG uses sorting and median selection technique for codebook generation. We have set of

feature points as shown in Eqn. 5.35. We apply KMCG & KFCG repetitively on Time axis of the feature points. In KFCG we cluster the codevectors and final codevector is taken as Column Average of the vectors. The steps are as follows.

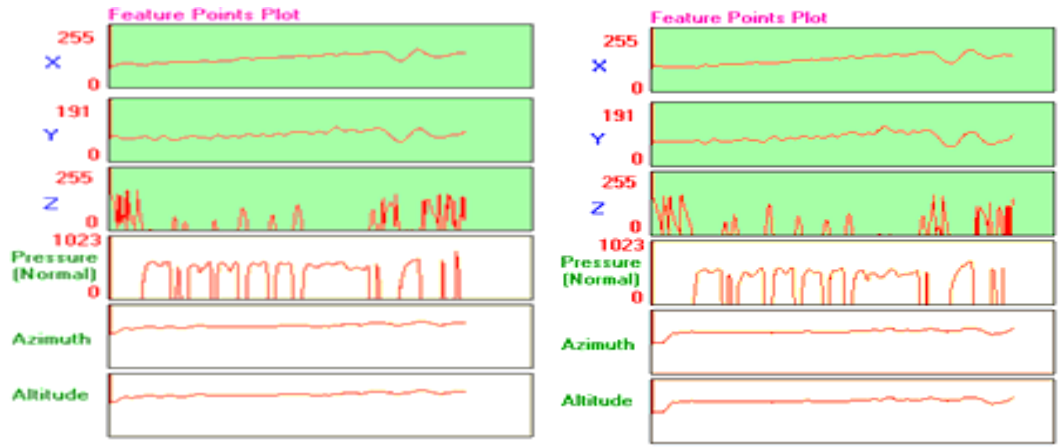
1. Capture the Signature points and store in an array.
2. Interpolate the feature points to generate maximum points on signature curve.
3. Divide the time interval of the signature into 'n' Intervals; this will be used for clustering the feature points.
4. For each interval read all the points sort the points in ascending order of the timestamp.
5. For KMCG, find median of the sorted array and consider it as Codevector for that interval. For KFCG we find the average of each column dimension e.g. X, Y, Z, Pr etc.
6. Repeat this procedure for all the intervals.
7. Copy each codevector in the codebook.

Actually we are implementing modified version of the KMCG & KFCG, as we are not repeating clustering on all the feature point dimensions, which is actually done in KMCG & KFCG, as were interested to from cluster on the time axis automatically all the points in a cluster tend to have closely matching features. If a signature is done in 4 seconds and we are using 100 clusters then each cluster contains points in 4ms vicinity on time axis. In our testing we have used 25 intervals for clustering, so the codebook has a size of 25 rows and 7 Columns [25X7]. We have generated two sets of codebook one with normal signature template (KFCG & KMCG) and one with interpolated template (KFCG-IP & KMCG-IP). We have compared this performance with normal KMCG & KFCG also. As discussed earlier for iris we apply KFCG & KMCG to the signature points (Eqn. 5.35) and generate normal KFCG & KMCG codebooks of size 128 also. These codebook acts as feature vectors. We use this codebook for enrollment and matching of signature. Signature matching can be performed by taking the Euclidian Distance (ED) between the codebooks. For signature of same user the ED is low, and for forged signature ED is high. We have to evaluate user specific thresholds for proper classification.

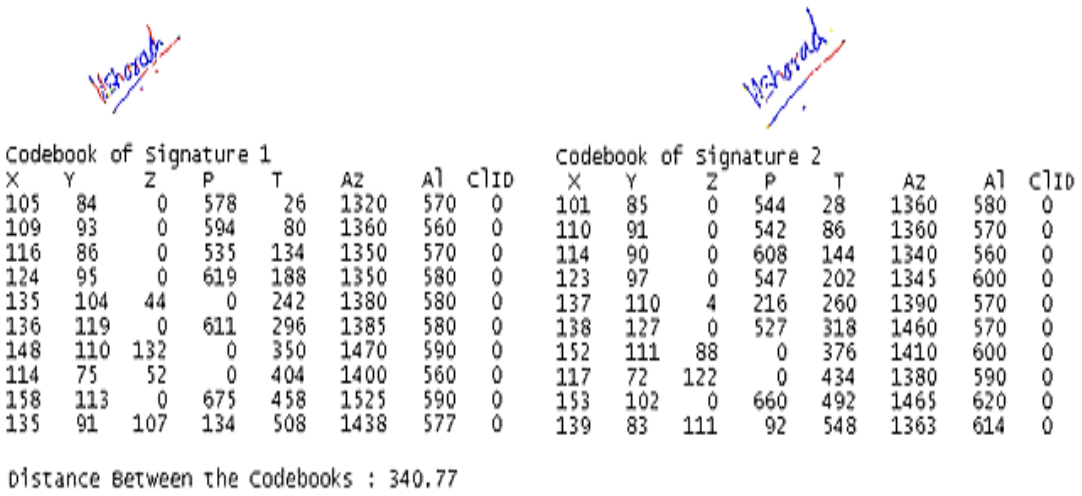
5.3.2 Enrollment & Testing

The extracted codebooks are stored in to database. The human signature is dependent on varying factors, the signature characteristics change with the psychological or mental condition of

a person, physical and practical condition like tip of the pen used for signature, signatures taken at different times, aging etc. This is captured in the codebook.



(a)



Codebook of Signature 1

X	Y	Z	P	T	Az	Al	ClID
105	84	0	578	26	1320	570	0
109	93	0	594	80	1360	560	0
116	86	0	535	134	1350	570	0
124	95	0	619	188	1350	580	0
135	104	44	0	242	1380	580	0
136	119	0	611	296	1385	580	0
148	110	132	0	350	1470	590	0
114	75	52	0	404	1400	560	0
158	113	0	675	458	1525	590	0
135	91	107	134	508	1438	577	0

Codebook of Signature 2

X	Y	Z	P	T	Az	Al	ClID
101	85	0	544	28	1360	580	0
110	91	0	542	86	1360	570	0
114	90	0	608	144	1340	560	0
123	97	0	547	202	1345	600	0
137	110	4	216	260	1390	570	0
138	127	0	527	318	1460	570	0
152	111	88	0	376	1410	600	0
117	72	122	0	434	1380	590	0
153	102	0	660	492	1465	620	0
139	83	111	92	548	1363	614	0

Distance Between the Codebooks : 340.77

(b)

Fig. 5.14. (a) Sample Signatures Their Dynamic Characteristics (b) Corresponding Codebook (Time KFCG) Snapshots

Two signatures from the same user & the dynamic characteristics are shown in the plots of Fig. 5.14. We apply Time based VQ using KMCG to generate two codebooks, each codebook is having feature vector with parameters X, Y, Z, Pressure, Timestamp, Azimuth, Altitude (CLID is the Cluster ID assigned –used for internal calculation), the corresponding Euclidian distance is 340.77 for the same users signature. For different user's signature or forged signatures the observed distance range is (900 to 2500). One typical cluster snapshot is also shown in Fig. 5.14.

5.3.3 Results for VQ Based Signature Recognition

The proposed technique was tested on 151 signatures collected from 25 different persons, some signatures were collected in one sitting, and some were collected in two sittings with a time gap of 2 to 7 days. Testing was performed on the signature template with and without interpolation. Total 300 tests were performed out of which 150 tests were performed for intra-class matching and 150 tests were performed for inter-class matching and forged signatures. The program was tested on AMD 64 1.8GHz, Windows XP SP3 and Visual C# 2005, .NET Framework 2.5.

Table 5.2
Performance Comparison for VQ Based Online Signature Recognition

VQ Algorithm	Normal Signature Template		Interpolated Signature Template	
	PI	CCR	PI	CCR
KFCG	86.40	85.00	93.50	92.00
KMCG	85.00	82.68	91.60	87.00
KFCG Time	93.40	91.25	97.00	95.00
KMCG Time	94.08	93.00	95.20	94.05

Table 5.2 shows summary of testing. Interpolated signature templates have given higher accuracy. Clustering along the time axis adds dynamic characteristics to the codebook and further boosts the performance of the VQ method. Best performance is given by KFCG on time axis applied to interpolated signature by giving PI of 97%, EER of 3.00% & CCR of 95%.

Fig. 5.15 shows FAR-FRR Plot for KFCG based feature vector for normal clustering. For normal signature template clustering achieved EER is 13.6% while that on interpolated signature template clustering gives better EER of 6.5%. Fig. 5.16 shows FAR-FRR Analysis of KMCG based feature vectors for normal clustering. For normal signature template clustering achieved EER is 15% while that on interpolated signature template clustering gives better EER of 8.40%. This performance improvement is due to proposed preprocessing by MDDA interpolation. Performance of KFCG is better in case of normal clustering.

In another approach we have used clustering on time axis to add dynamic characteristics to the codebook. Time KFCG performs well by achieving PI of 93.40% and 97% with and without interpolation based preprocessing, Fig. 5.17 & Fig. 5.18 show FAR-FRR plot for this. Fig. 5.17 & Fig. 5.18 show corresponding plot for Time KFCG &

Time KMCG. Fig. 5.19 shows comparison for all these variants. We can clearly see that the clustering across time axis gives better results as it captures dynamic nature of the signature.

Interpolation based preprocessing by Modified DDA algorithm boosts the performance by reducing sampling error. Maximum testing time per signature was 20ms this make the system feasible for real time application.

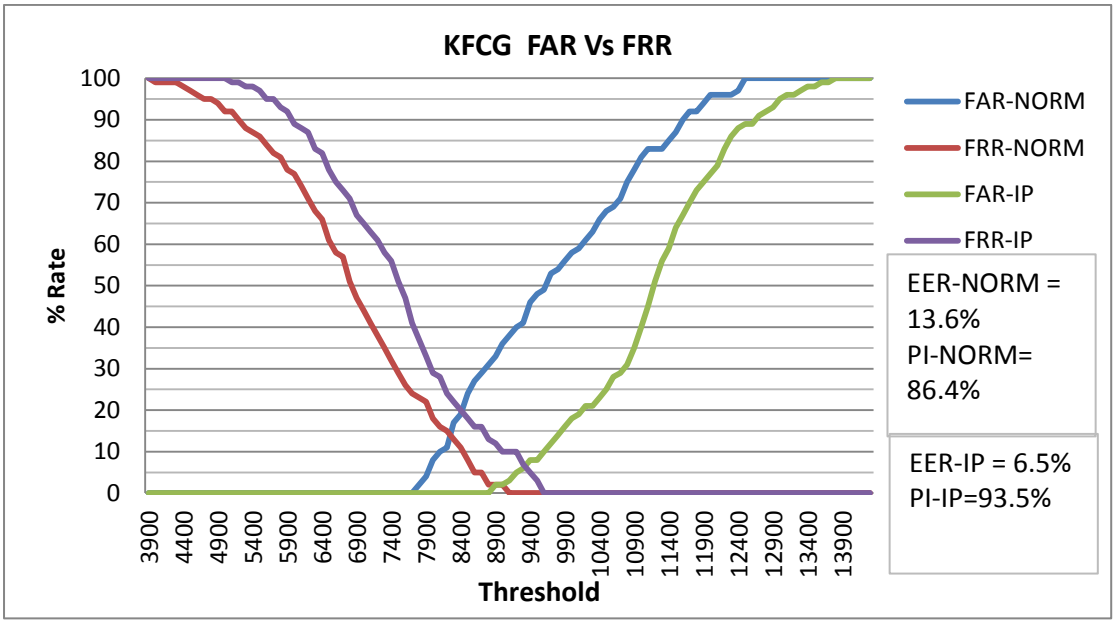


Fig. 5.15. FAR-FRR Analysis of KFCG based Feature Vectors for Normal Clustering

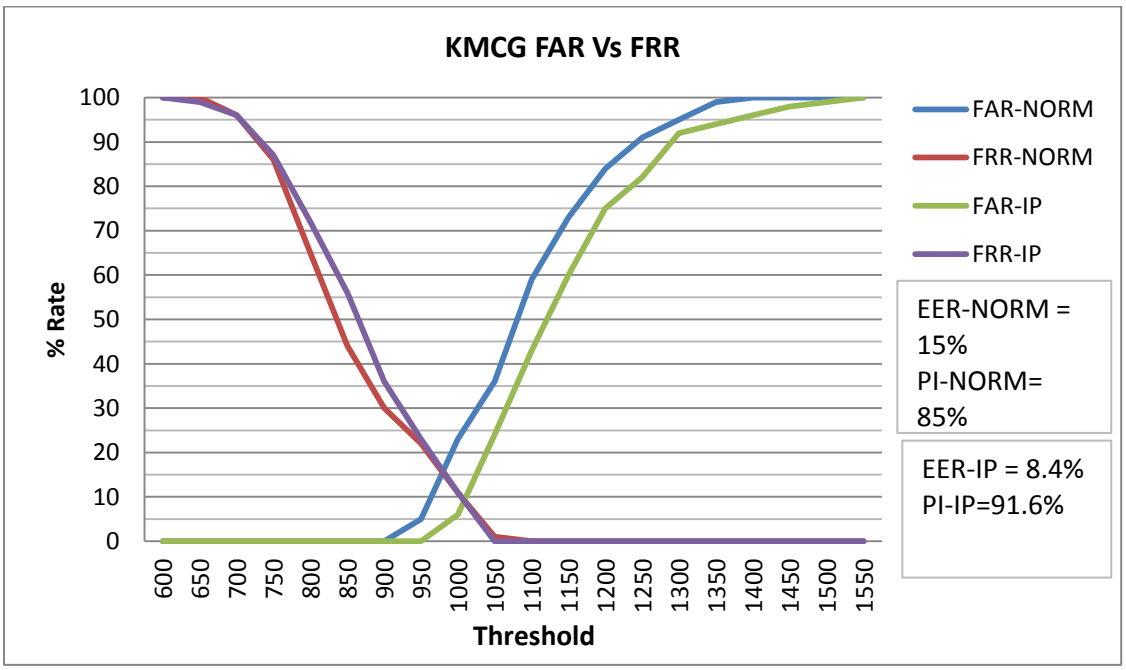


Fig. 5.16. FAR-FRR Analysis of KMCG based Feature Vectors for Normal Clustering

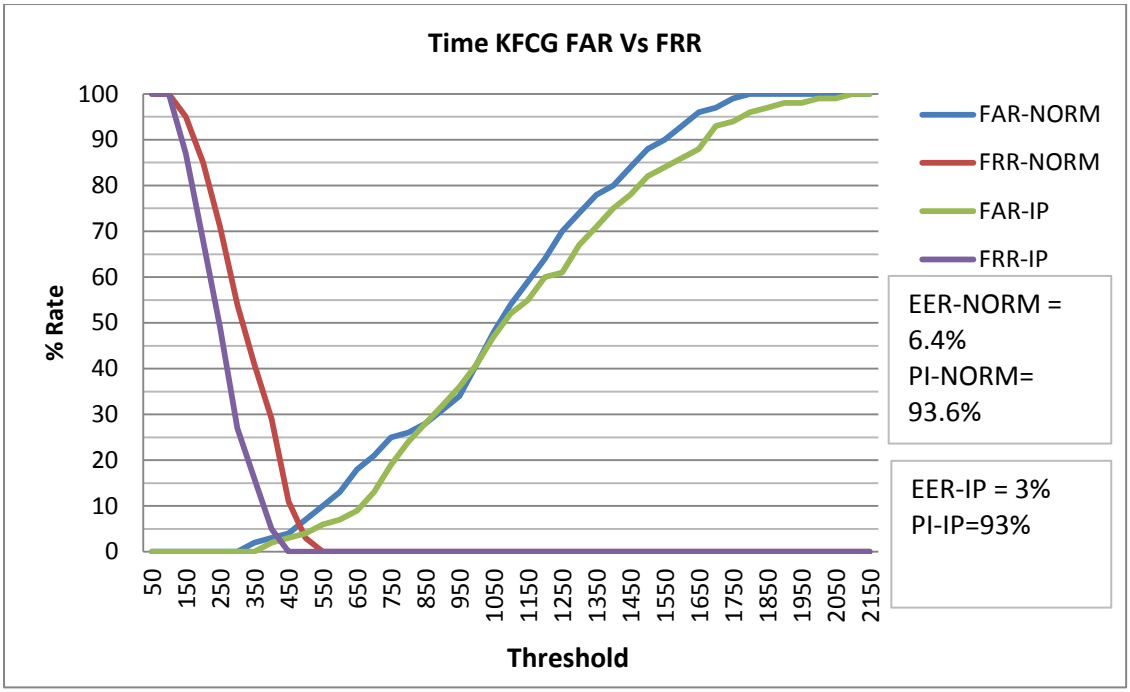


Fig. 5.17. FAR-FRR Analysis of Time KMCG based Feature Vectors for Time Axis Clustering

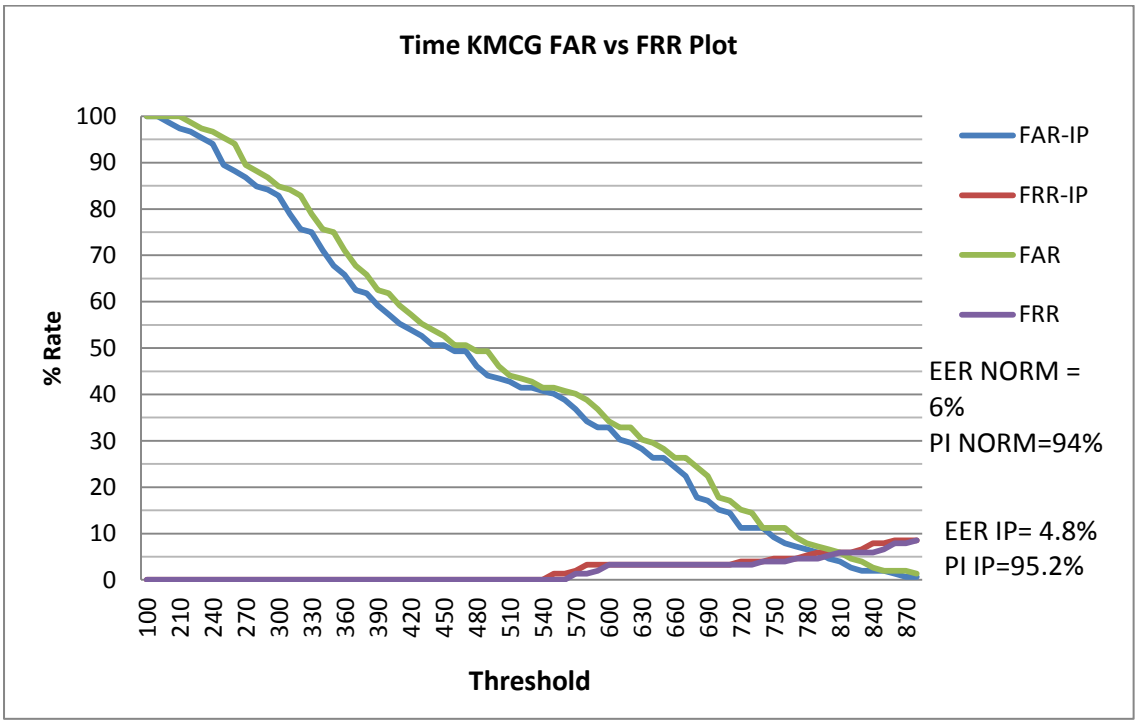
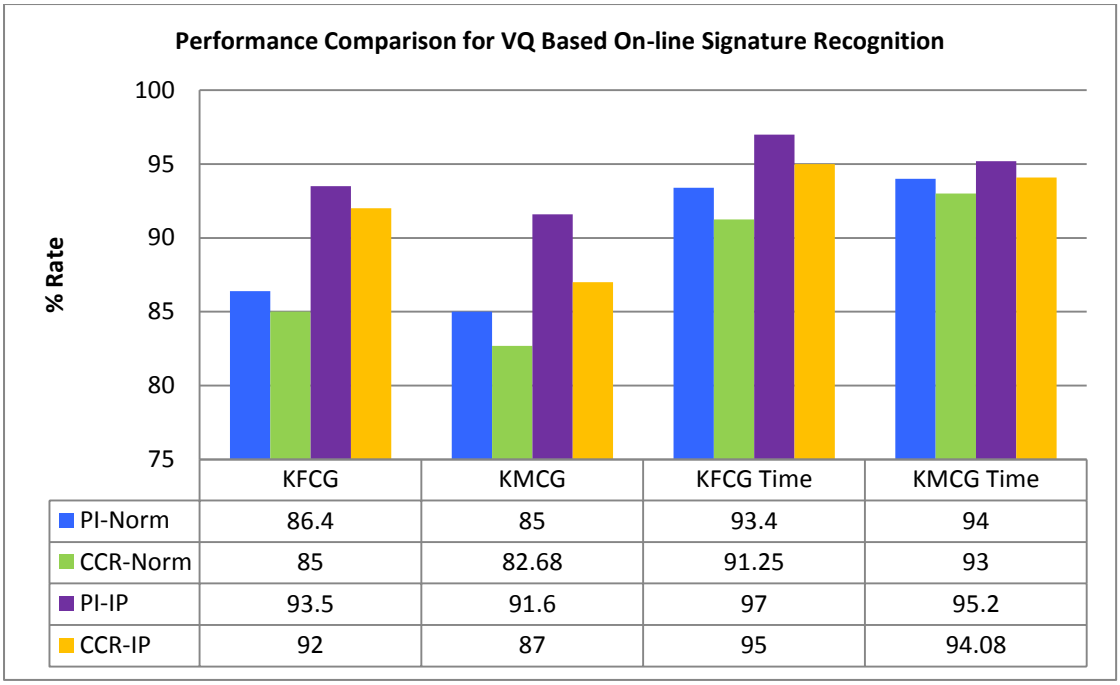


Fig. 5.18. FAR-FRR Analysis of KMCG based Feature Vectors for Time Axis Clustering

In this section we have discussed KFCG & KMCG based feature vectors. Normal clustering as well as time axis clustering is performed to generate VQ Codebook based feature vectors. In the next section we continue our discussion on dynamic signature recognition using transforms.



	KFCG	KMCG	KFCG Time	KMCG Time
PI-Norm	86.4	85	93.4	94
CCR-Norm	85	82.68	91.25	93
PI-IP	93.5	91.6	97	95.2
CCR-IP	92	87	95	94.08

Fig. 5.19. Performance Comparison of KFCG & KMCG based Feature Vectors

5.4 Online Signature Recognition Using Transforms

In this section we use one of the widely used signal processing techniques. We are using 'Transforms' to extract spectral features from the dynamic signature feature points. We are considering derived features such as velocity, acceleration & their angles for this calculation. Here we are discussing multi-algorithmic biometric using multiple features. We have used Discreet Cosine Transform (DCT), Discrete Fourier Transform (DFT), Walsh Hadamard Transform (WHT), Kekre's Transform (KT). The transform are used on following feature vectors.

1. Signature Feature Points $P_i=\{X_i,Y_i,Z_i,T_i,Pr_i,Az_i,Alt_i\}$.
2. Fusion of Velocity and Acceleration Feature points.
3. Row & Column Mean of the pressure distribution of the Signature.
4. Row & Column Standard Deviation of the pressure. distribution of the signature pressure distribution.

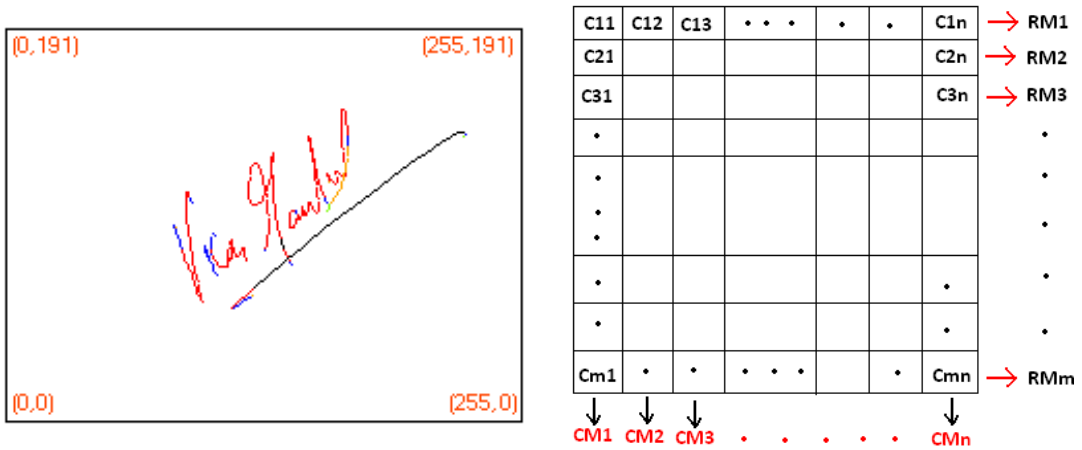
We have already discussed Signature Feature Points based feature vectors. We briefly discuss option 2, 3 & 4 in the next part.

5.4.1. Velocity & Acceleration Based Feature Vector

We have the captured signature points for the signature. These points have dynamic signature information as discussed previously. We derive velocity and acceleration information from the point as discussed in section 5.4.2, we derive Velocity points V_i (Velocity Magnitude) and Acceleration Ai (Magnitude) from Pi. If we have scanned 'n' points of P_i then for Velocity & Acceleration we have n/4 points. We apply 1D transform on these points to generate the feature vector. The transform coefficients are used as feature vector.

5.4.2 Row, Column Mean Based Feature Points

We have used the concept of Row & Column mean in case of iris recognition. This is a very good feature vector for extracting texture information. We have used this in Section 4.2.2 for iris recognition using WHT & DCT. We will use this method on the pressure distribution of the signature. We normalize the pressure distribution map by matching the center of gravity of the signature with the center of 256*192 pixel window. We find the row wise & column wise mean. The pressure map is shown in Fig. 5.20(a) and Row mean and column mean generation is shown in Fig. 5.20(b). We get two vectors for mean. We normalize them by padding zeros and make every vector 256 points long.



(a) **(b)**

Fig. 5.20. (a) Dynamic Signature Pressure Information P[256,192] (b) Generation of Row Mean (RM) & Column Mean (CM) vector From Signature Pressure Map P_{ij}

We take transforms (DCT, WHT, DFT, and KT) of these row & column mean vectors. These vectors contain both pressure and texture information of the signature. We use the transform coefficients as feature vector of the signature. For all these testing

we are using the interpolated signature template as it improves the accuracy. In the next section we discuss the results. The fusion of feature vectors based on signature features, velocity & acceleration, row & column mean is also performed.

5.4.3 Results for Transforms Based Signature Recognition

The testing is performed on 248 different signatures collected from 31 people's signatures. Volunteers were asked to perform different levels of forgeries including random & skilled forgery [1], [3]. Total 1676 different tests were performed on above mentioned 4 different feature vectors. The summary is given in Table 5.3 & Fig. 5.21.

Table 5.3
Performance Comparison for Velocity, Acceleration, Row Mean & Column Mean Feature Vector for Online Signature Recognition

Transform	Signature Features		Velocity & Acceleration		Row & Col. Mean		Fusion	
	PI	CCR	PI	CCR	PI	CCR	PI	CCR
DCT	76.00	74.33	52.01	50.00	69.00	68.22	78.00	77.00
DFT	77.00	75.02	64.00	61.22	76.00	74.78	79.00	78.00
WHT	78.00	76.88	58.00	55.63	69.00	65.36	79.00	77.56
KT	79.00	78.87	58.01	57.33	72.00	71.22	80.00	77.88

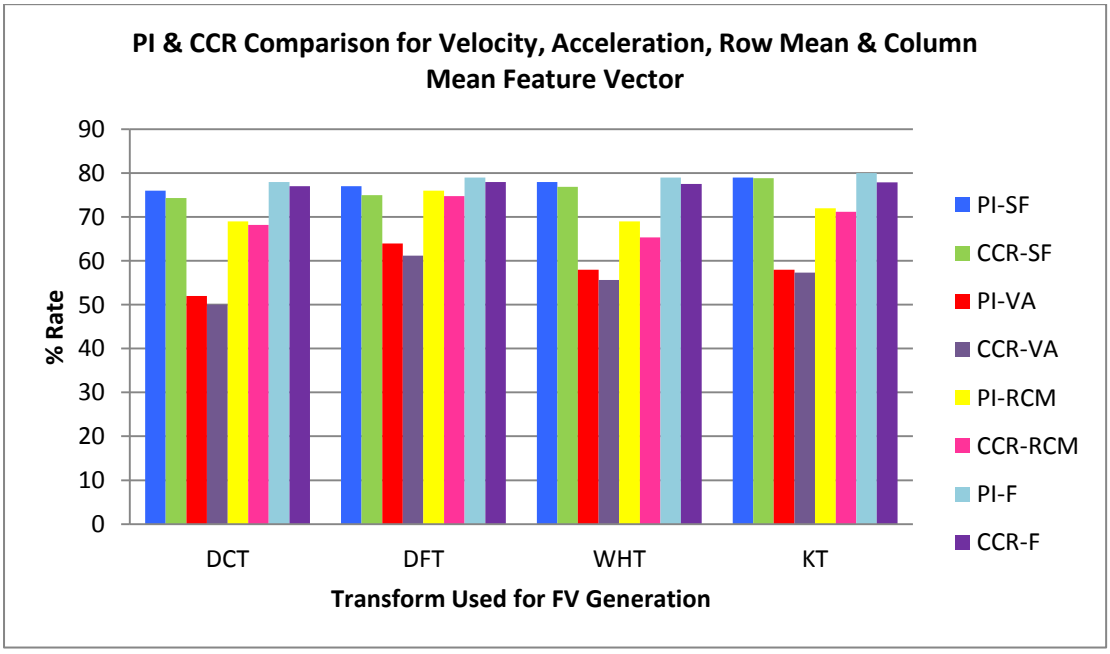


Fig. 5.21. Performance Comparison for PI & CCR of Velocity, Acceleration, Row Mean & Column Mean Feature Vector for Online Signature Recognition

(Feature Vector Types: SF-Signature Features, VA- Velocity & Acceleration, RCM- Row & Column Mean, F-Fusion)

We have tested Signature Features (X, Y, Z, Pressure, Azimuth Altitude) , Combination of Velocity & Acceleration, Row & Column Mean & their fusion separately. The results have following highlights

1. Signature Features (X, Y, Z, Pressure, Azimuth, and Altitude) give best Equal Error rates.
2. Amongst the transform DFT & KT has highest CCR, and other transforms perform marginally lower.
3. Row & Column Mean based feature vectors have good performance.
4. Performance of velocity & acceleration based feature vector is lowest.
5. Fusion of feature vector gives improved performance. This is an example of feature fusion (to form a multi-algorithmic biometric system).

Next we present summary of all the dynamic signature recognition methods discussed in these chapter.

Table 5.4
Performance Comparison for Different Dynamic Signature Recognition Methods

Sr.	Feature Vector Type	Performance Metrics	
		PI	CCR
1	Gabor Normal	92.00	90.25
2	Gabor Timestamp	95.20	94.00
3	Gabor DTW	82.00	77.00
4	KFCG	86.40	85.00
5	KMCG	85.00	82.68
6	KFCG-Time	93.40	91.25
7	KMCG-Time	94.00	93.00
8	KFCG-IP	93.50	92.00
9	KMCG-IP	91.60	87.00
10	KFCG-Time-IP	**97.00**	**95.00**
11	KMCG-Time-IP	95.20	94.08
12	DCT	78.00	77.00
13	DFT	79.00	78.00
14	WHT	79.00	77.56
15	KT	80.00	77.88

We have tested total 15 different feature vector extraction methods. They are listed in Table 5.4. Vector Quantization using KFCG along time axis on interpolated signature points gives best performance by achieving 97 % PI & 95% CCR.

The Proposed preprocessing technique based on Modified DDA based interpolation improves performance significantly by boosting the CCR in the range of 1 to 7%. Fig. 5.22 shows comparison of EER for TAR-TRR analysis of Dynamic SRS discussed here.

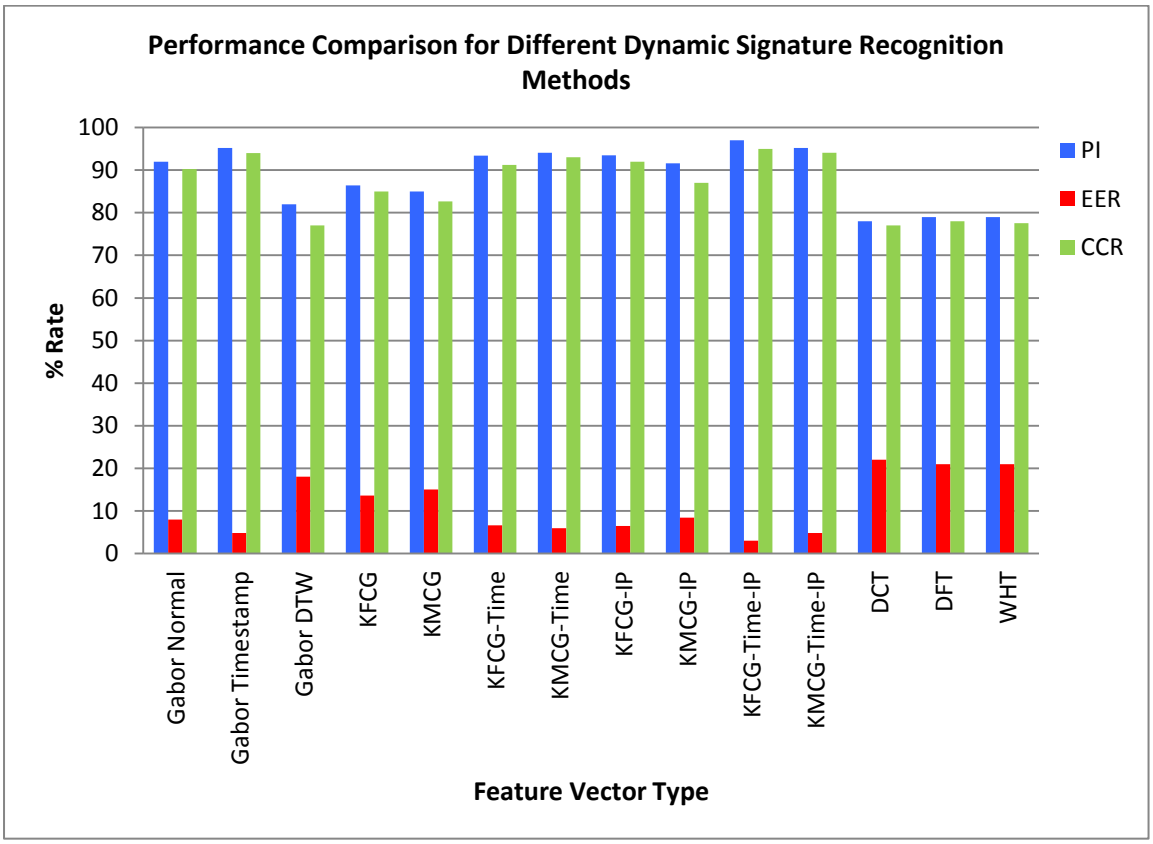


Fig. 5.22. Performance Comparison Dynamic SRS Methods

In this section we have studied dynamic handwritten signature as biometric traits for authentication. They are also classified as behavioral biometrics or biometrics based on human behavior. In the next section we present a brief study about keystroke dynamics which is another of such type.

5.5 Study of Keystroke Dynamics (KD) as a Biometric Trait

Keystroke dynamics, or typing dynamics, is the detailed timing information that describes exactly when each key was pressed and when it was released as a person is typing at a computer keyboard [263]-[266]. Keystroke dynamics is a safeguard based on authenticating access to computers by recognizing certain unique and habitual patterns in a user's typing rhythm [268].

It is argued that [263] the use of keystroke rhythm is a natural choice for computer security. This argument stems from observations that similar neuro-physiological factors that make written signatures unique and they are also exhibited in a user's typing pattern [267]. When a person types, the latencies between successive keystrokes, keystroke durations, finger placement and applied pressure on the keys can be used to construct a unique signature (i.e., profile) for that individual. For well-known, regularly typed strings, such signatures can be quite consistent. Furthermore, recognition based on typing rhythm is not intrusive, making it quite applicable to computer access security as users will be typing at the keyboard anyway. In this section we discuss keystroke dynamic information extraction and matching. We are using keystroke timing information to generate probability distribution for the password; this information is used for matching the password keystroke sequence containing timing information.

As a person types, the KD application collects the duration of each key press and the cycle time between one key press and the next. Once the password is typed we calculate the information content of the password. This is done by calculating the entropy of the obtained timing values.

A. Entropy [216] is a measure of the uncertainty associated with a random variable. The term by itself in this context usually refers to the Shannon entropy [216], which quantifies in the sense of an expected value, the information contained in a message. This is usually in units such as bits.

The entropy H of a discrete random variable X with possible values $\{x_1, \ldots, x_n\}$ is

$$H(X) = E(I(X)) \tag{5.24}$$

Here 'E' is the expected value, and 'I' is the information content of X. $I(X)$ is itself a random variable. If 'P' denotes

the probability mass function of '*X*' then the entropy can explicitly be written as

$$H(X) = \sum_{i=i}^{n} P(x_i) I(x_i) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i) \qquad (5.25)$$

Where '*b*' is the base of the logarithm used base=2.
For verification purposes a known verification string is usually typed (i.e. account ID and password). Once the verification string is entered, it is processed by an algorithm that compares the person's typing behavior to a sample collected in a previous session. The comparison is made by calculating the relative entropies of the sample and the test.

B. Relative Entropy [216] - Suppose the probabilities of a finite sequence of events is given by the probability distribution $P = \{p_1...p_n\}$, but somehow we mistakenly assumed it to be $Q = \{q_1...q_n\}$. According to this erroneous assumption, our uncertainty about the j^{th} event, or equivalently, the amount of information provided after observing the j^{th} event,
The (assumed) average uncertainty of all possible events is then

$$-\sum_j p_j \log q_j \qquad (5.26)$$

On the other hand, the Shannon entropy of the probability distribution *p*, defined by,

$$-\sum_j p_j \log p_j \qquad (5.27)$$

is the real amount of uncertainty before observation. Therefore the difference between these two quantities

$$-\sum_j p_j \log q_j - (-\sum_j p_j \log p_j) = \sum_j p_j \log p_j - \sum_j p_j \log q_j \qquad (5.28)$$

is a measure of the distinguishability of the two probability distributions *p* and *q*. This is precisely the classical relative entropy, or Kullback–Leibler divergence:

$$D_{KL}(P \| Q) = \sum_j p_j \log \frac{p_j}{q_j} \qquad (5.29)$$

The output of the comparison is a score. If this is the first time the KD system has seen this user, the results of this process are used to enroll him instead of verifying his identity. This distance has been used previously in matching wavelet energy sequence; here this

metric is used for matching password keystroke's timing duration sequences.

5.5.1 Capturing Keystroke's Timing Information

The password consists of characters. These characters are entered through keyboard, when a key is pressed an event is raised. Visual Studio 2008 is used for programming, in VS 2008 the related events are keydown, keyup & keypressed. We use these events to extract timing information of the password.

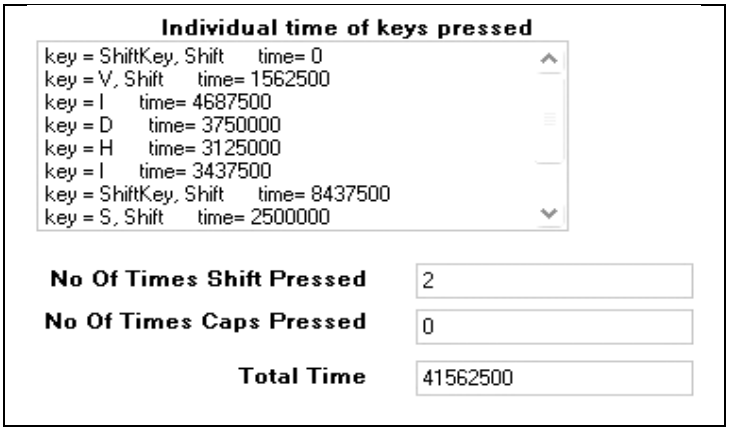


Fig. 5.23. Captured Keystroke Data for Password "VIDHIS"

The captured time is in a composite format called as 'Ticks' in VS 2005 time units. One such capture event is shown in Fig. 5.23. The captured data is the time instance when the key is down, we get the keypress time & this data is normalized by converting into milliseconds & dividing each time value by total time. In this way timing information of different events 'E' is generated. $E=\{E_1,E_2,,\ldots..,E_n\}$, Where, E_i is i^{th} key in the password.

TE_i= (keypress Time)$_i$ + (keydown Time)$_i$ + (Flight Time)$_i$ (5.30)

The total time required to type the password is 'T_{total}' given by,

$$T_{total} = \sum_{i=1}^{n} TE_i$$

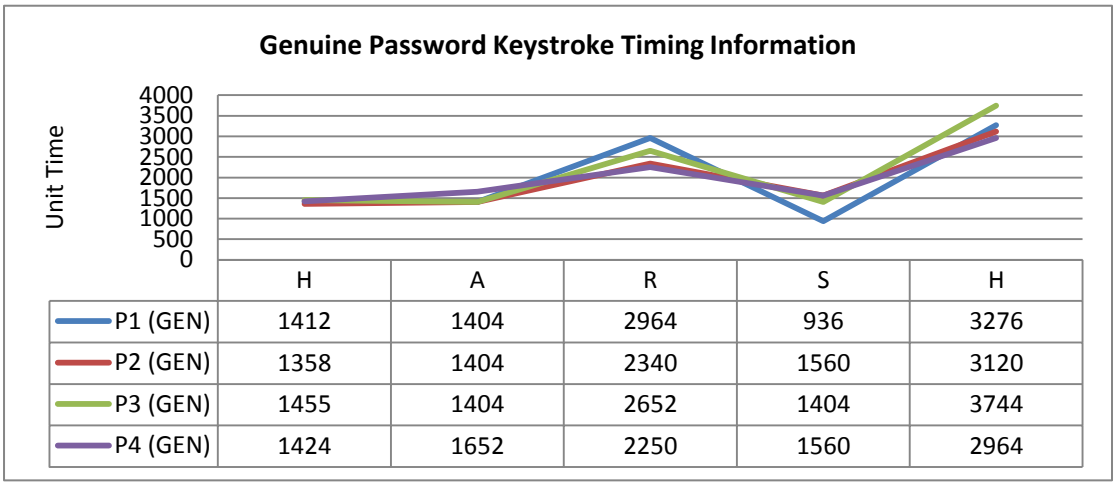
(5.31)

From this information we generated the normalized PDF (Or Probability Mass Function) for the password 'P' as P= {P_1, P_2, P_3, ….., P_n}.

$$P_i = \frac{TE_i}{T_{total}} \qquad (5.32)$$

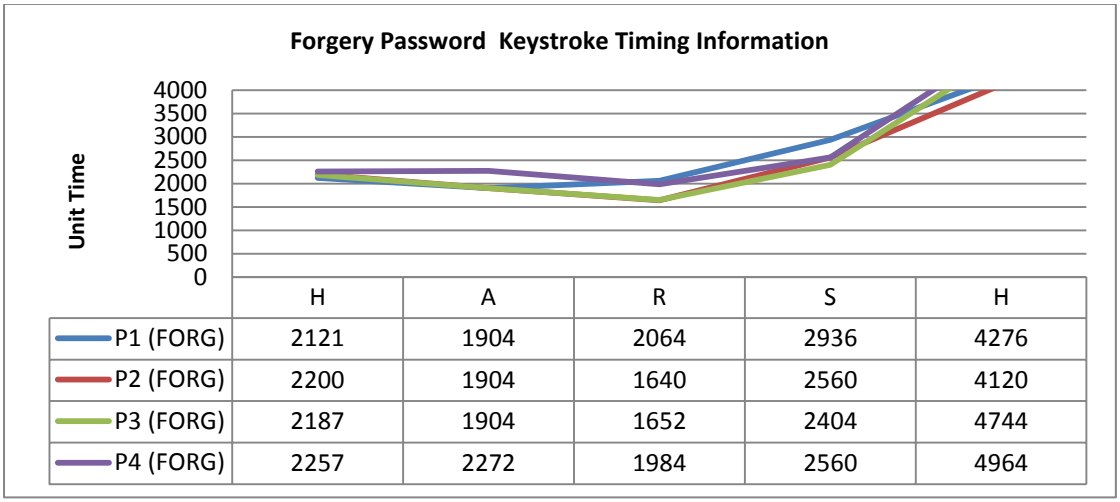
This normalized information is then used for password matching. The timing information for Genuine & Forged password (HARSH) is

shown in Fig.5.24 (a) & (b). The passwords are scanned in one sitting hence it can be seen that the timing pattern for genuine & forgery is different. To match two password sequences Pi & Qi, Relative Entropy as discussed in this section as well as Euclidian distance between two timing sequences has been used.



	H	A	R	S	H
P1 (GEN)	1412	1404	2964	936	3276
P2 (GEN)	1358	1404	2340	1560	3120
P3 (GEN)	1455	1404	2652	1404	3744
P4 (GEN)	1424	1652	2250	1560	2964

(a)



	H	A	R	S	H
P1 (FORG)	2121	1904	2064	2936	4276
P2 (FORG)	2200	1904	1640	2560	4120
P3 (FORG)	2187	1904	1652	2404	4744
P4 (FORG)	2257	2272	1984	2560	4964

(b)

Fig. 5.24. Captured Keystroke Data for Password "HARSH". Showing Plots for Genuine & Forged Password Keystroke Timings.

(Timing Information is shown in "Ticks" i.e. Timing Units for Keystroke Events in MS Visual Studio 2008. This Graph is for Representation.)

5.5.2. Results for Keystroke Dynamics

This method is implemented using Microsoft Visual Studio 2008, & tested on Intel Core2Duo 2.4 GHz, 2GB RAM Running on Windows Vista Operating System. Ten passwords each from 33 different users were collected; the passwords were 4 to 6 characters long. For classifying passwords Euclidian distance between normalize pdf

& relative entropy is used. Total 3325 Genuine & 1320 Forgery tests were performed for Distance Analysis.

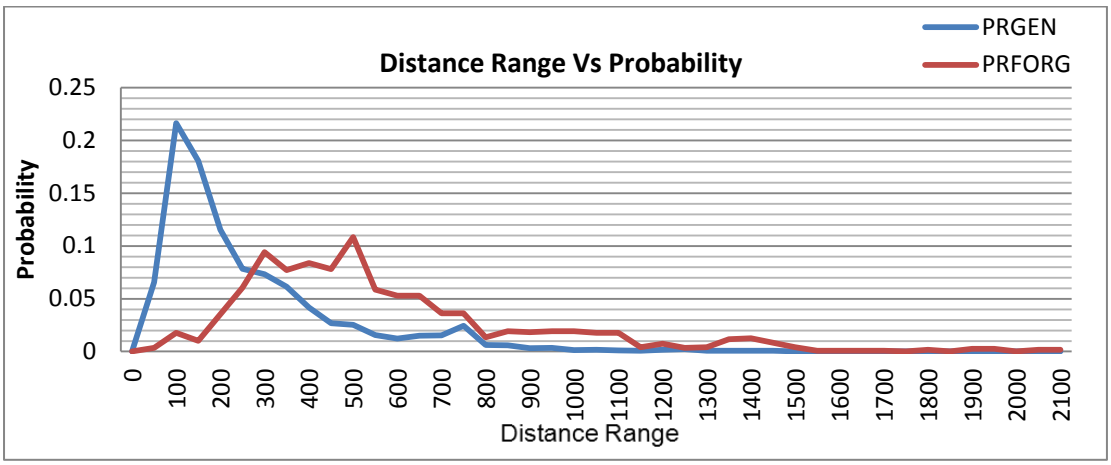
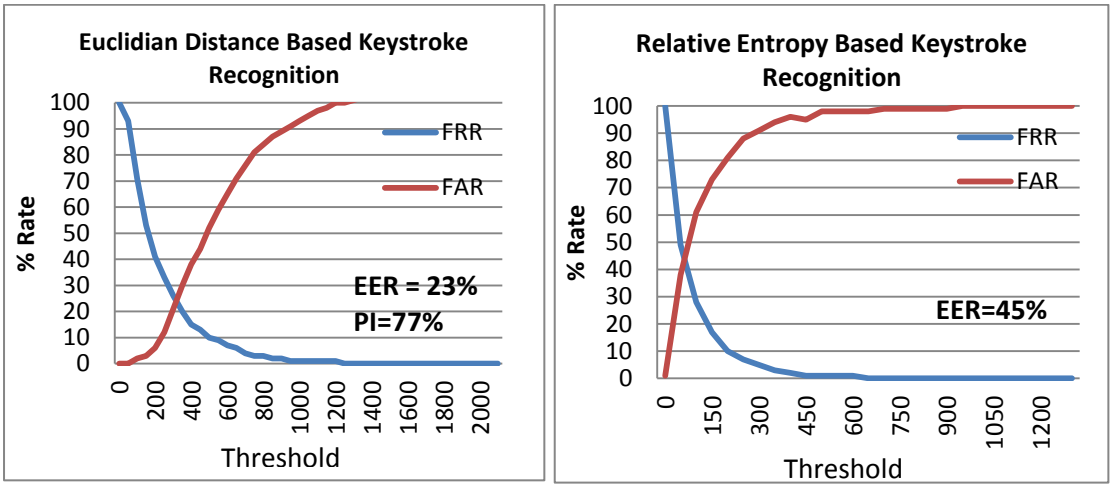


Fig. 5.25. Distance Range vs. Probability. Two Peaks Corresponding to Genuine (PRGEN) & Forgery (PRFORG) are Clearly Visible.

Fig. 5.25 shows distance vs. its probability of occurrence, this plot is for the Euclidian distance between the password timing sequences. This clearly indicates that the password timing information can be used for classification. One more thing that should be noted is this is pure distance and not like 'Trigraph' & 'Bigraph' Sequences [268]. If we use this metrics then the separation is higher.



(a) **(b)**

Fig. 5.26. FAR-FRR Analysis for (a) Euclidian Distance Based Keystroke Recognition (b) Relative Entropy Based Keystroke Recognition.

FAR-FRR analysis is shown in Fig. 5.26 (a) & (b). Euclidian distance based classifier performs better giving EER of 77%, Relative Entropy based classifier shows poor performance by giving EER=55%; the Correct Classification Ratios (CCR) are 75% & 53% respectively at a distance threshold of 370. Here simple K-NN

classifier is used, with only 5 samples for training. The performance can be improved by considering longer password length of 6-8 characters and the use of special characters. The study shows the keystroke dynamics is viable option for computer security; it can be used to strengthen the password based authentication programs.

5.6 Summary

In this chapter two important biometrics based on human behavior have been studied. Handwritten Signature & keystroke dynamics are the emerging behavior based biometrics and with the advancements in hardware we can capture dynamic information easily. The handwritten signature recognition systems (SRS) in online mode have been discussed in detail. Online signature recognition systems need specific hardware for capturing dynamic information.

The captured data is processed by a novel scheme based on Modified Digital Difference Analyzer Algorithm (MDDA) proposed in section 5.1.2. Signature templates are preprocessed by this algorithm and interpolation of captured data. Clustering algorithms such as KFCG, KMCG as well as Gabor filter based feature vectors are used for feature vector generation. KFCG on time axis gives best performance for signature recognition.

Transforms are also used on velocity and acceleration based feature points. We have discussed multi-algorithmic online signature recognition system based on fusion of velocity and acceleration features as well as signature points in 3D-space (X, Y & Z co-ordinates in captured signature data packets). Orthogonal Transforms such as DCT, DFT, WHT and Kekre's Transform are tested. Finally another behavior based biometric 'keystroke dynamics' is discussed is section 5.5. Euclidian distance between keystroke timing sequence as well as relative entropy between normalized passwords timing PDF is used for classification.

Up till now in previous chapters fingerprints, palmprint, finger-knuckle print, face, iris, static as well as dynamic handwritten signature and keystroke dynamics has been discussed in detail. Various unimodal aspects are mainly discussed but multi-algorithmic variation for some of the biometrics are also discussed. In the next chapter multimodal biometric are discussed in more detail along with this fusion of feature vectors is also discussed as it is an important process.