University of Sussex, UK

Real-time hand gesture recognition for small devices

Thesis,
MSc in Evolutionary and Adaptive Systems

Rudra PK Poudel
rudrapoudel@gmail.com

September, 2009

Acknowledgement

First to many people who helped me and contributed to shape this thesis, I owe them all a debt of gratitude. Those listed below deserve special mention and thanks.

My sincere thanks to Dr. David Young, my supervisor, who supervised my work from the very beginning – from collection of vague ideas to the completion of the project as it stands now; for his guidance, advice, and criticisms during this work and moreover, his encouragement when I feel low.

My thanks go to my colleagues at University Lab, who always use to with me whether I need to go for walk at midnight or food when I am hungry. Moreover, never bored to see my unsuccessful experiments and encourage me each day for more hard work.

My special thanks to my lovely mom, Mandhari Poudel, who has patience to stay alone without seeing me and supporting me from every angle to complete this dissertation as well as this degree. Last but definitely not least, I would like to thank all individuals who directly or indirectly helped me in this work.

Abstract: This dissertation proposes a vision based hand gestures recognition system for low resource devices using a temporal template based approach. The five most popular machine learning algorithms- Naïve Bayes (NB), Radial Basis Function (RBF), Multi-Layer Perceptron (MLP), SVM and sparse Bayesian classifier or Relevance Vector Machine (RVM) are used to test the viability of the methods for low resource devices. It is found that RBF and NB outperform the other methods. Correct classification rates of 85% and 81.22% are obtained from RBF and NB in a loosely controlled noisy environment. These results are not only comparable to the best methods available but are also suitable for implementation in real-time environments and on low resource devices.

Contents

Acronym

HCI	Human Computer Interaction
HMI	Human Machine Interface
HMM	Hidden Markov Model
K	Size of features vector or used most variant principal components or number of features
MEI	Motion Energy Image
MGO	Motion Gradient Orientation
MHI	Motion History Image
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
RBF	Radial Basis Function
RVM	Relevance Vector Machine or sparse Bayesian classifier
SIFT	Scale-Invariant Features Transform
SVM	Support Vector Machine
tMHI	timed Motion History Image

1 Introduction

1. Introduction

Hands are an intuitive way to communicate in human-to-human interactions. Hand communications are even more important for the hearing impaired. Hence, hand gestures would be a more natural and effective way to communicate with robots, computers, mobiles and other machines than using devices like keyboards, mice, touch panels and joysticks. Though Artificial Intelligent (AI), especially machine vision and machine learning, has progressed a lot in the last two decades the way humans communicate with machines has remained largely unchanged. Moreover, the decreasing prices and increasing power of electronic devices like computers and cameras are another attractive reason for more attention to be paid to vision based human machine interaction.

In the last few years, hand gestures recognition has received much attention. Many review papers of hand gesture recognition have been published, for example Pavlovic, Sharma and Huang (1997), Wu and Huang (1999), Hassanpour, Wong and Shahbahrami (2008), Garg, Aggarwal and Sofat (2009) and others are evidence of increasing research interest in hand gesture recognition. Starner, Weaver and Pentland (1998) are able to correctly recognize 98% of American Sign Language in a restricted environment, which was a promising result at that time. However, even one decade after the early success of Starner, Weaver and

Pentland (1998), none of the existing methods has achieved 100% correct classification, except with a much lesser number of simple gestures in a restricted environment. Achieving a good success rate with a small training set and complex gestures in a non-restricted environment is still out of the reach of existing systems. Moreover, gesture recognition in real-time is still a big challenge due to processing complexity.

Self-occlusion of fingers, color variation due to environmental changes, personal variation of hand shape and movement and variation of hand motion due to fatigue are all very difficult problems for hand gesture recognition. Moreover, the hand has more than 25 degrees of freedom (Francke, Ruiz-del-Solar and Verschae, 2007), which cause the lots of variation on hand gestures movement. Hence, this dissertation will explore whether an easy task for a human is really a difficult task for an intelligent system or whether we could solve the problem using a simple adaptive method. Moreover, the aim of this research work is to understand the problems of hand gesture recognition in more detail. Alternative approaches to solving the problem will be experimented with along with discussions of the possible ways forward for this complex problem of machine vision as well as machine learning. This dissertation will experiment with 9 hand gestures ranging from simple to complex.

In the past, researchers had used different kinds of additional hand devices to recognize the hand gesture, such as mechanical gloves (Fels and Hinton 1997) and marker in figure tips (Kim, Albuquerque and Havemann 2004). However, additional devices increase the cost of the system and creates uneasiness while making the hand gesture as well as being an unnatural way of communication. Alternatively, vision based methods are less difficult to implement and easy to use. Hence, this dissertation will focus on vision based hand gesture recognition approach. Also, it is worth to mention here that gestures are also related with other parts of the body other than the hand, like body gestures which is especially use to human behavior analysis as well as interaction with virtual world.

Most of the hand gesture recognition models use Hidden Markov Model (HMM) as HMM achieves remarkable success in similar kinds of speech and handwriting recognition problems. HMM and their various extensions have been tried by many researchers like Parallel HMM (PHMM) (Vogler and Metaxas 2001), self-organizing HMM (Baure and Kraiss 2002), adaptive extensions of HMM (Wilson and Bobick 2000) and pseudo two dimension HHM (P2-HMM). However, HMM has been criticized recently because HMM requires a large number of training sets and HMM analyses each sign as a whole i.e. without breaking it into small meaningful parts (Wong and Cipolla, 2005). Therefore, this dissertation explores hand gesture recognition using the following methods: naïve Bayesian (NB) classifier, sparse Bayesian classifier also known as Relevance Vector Machine (RVM), Radial Basis Function (RBF), Multi-Layer Perceptron (MLP) and Support Vector Machine (SVM).

This dissertation will outline similar work as well as different approaches to hand gesture recognition in section 2. Research methodologies are described in section 3. In section 4, experimental results are described and in section 5, discussion of the results, limitations of the study, future work along with conclusion are described.

1.1 Research Question

Human-robot, human-computer and human-other machines interaction can all benefit from using hand signs as a more intuitive communication mechanism. However, most of the researches on the past have focused on increasing the success rate of hand gesture recognition and only a few of them considered the problems of real-time processing capacity, small training sets and minimum training time. More importantly, in the past there has been no focus given for small devices with low processing capacity, low memory and low resolution cameras. Hence, I strongly believe that we would be highly benefitted by a hand gesture recognition method available for low capacity devices. A method able to work on low

resource devices would of course be automatically able to work on high capacity devices.

Hence, this dissertation will explore the real-time hand gesture recognition methodologies for the small devices. It will focus on different methodologies and their efficiency rather than simply improving one methodology of choice. It is important to first discover which methodology is the most suitable for low resource devices. Also, it is worth mentioning here that there has been no previous work which compares the different methodologies in the same experimental setting.

2 Literature Review

2. Literature Review

There are many forms of human gestures, such as hand gesture, general body gesture and facial expression (Derpanis 2004). Sometimes only one hand is used for gestures and sometimes both hands are used. Also, sometimes hand gestures are used to express additional information along with verbal communication, while hand gestures are the only means of communication for deaf people. Therefore, some of the hand gestures are simple while others are complex. However, this dissertation will be focus on human machine interaction using single hand gesture.

Hand gestures are purposeful movement of the hand (Hassanpour, Wong and Shahbahrami 2008), which carries the meaning. According to psychological study, hand gesture consists three phases: *preparation*, *nucleus*, and *retraction*. *Preparation* phase occurs before the *nucleus* phase, it might be short as well as long. Major hand movement phase where the actual gesture occurs is called *nucleus* phase. *Retraction* is the end phase of the gesture i.e. termination of the gesture, which might not occur if another gesture follows on continuously.

According to bionics view, the understanding of the gestures depends upon effective tracking of the object of interest (i.e. the hand here in this

case) and not merely on visual information of the whole environment (Wang, Zhang and Dai 2007). Hence, it is very important that we must effectively handle the environmental noise i.e. unnecessarily movement except the hand in this case.

In general, vision based approaches fall into two different categories. One is the model based approach and another is the view based approach, which are separately summarized below.

2.1. Model Based Approach

The model based approaches tries to infer the knowledge of the hand posture using a 3D model of the hand skeleton. One of the earliest model based approach was proposed by Rehg and Kanade (1994), this uses the bare hand for gesture recognition. 3D models actually try to capture the information of the joints angles of the fingers and palm using multiple cameras to model the hand gesture. This kind of model is idealistic to communicate with virtual environment (Derpanis 2004).

The model based approaches tries to estimate the hand parameters (joint angle of fingers and palm) for hand tracking using 2D image frames captured using multiple cameras. Normally, 2 to 4 cameras are used depending upon complexity of the environment and gestures. The hand parameters estimation from the 2D images is an inverse mapping problem, which is non-linear due to the 3D mapping of 2D projected image frames. Though there are several methods that exist for optimum parameter estimation in such situation such as Newton's method but there is very high chance to get stuck in local optimum solution rather than the global optimum solution. Hence, estimation of best parameters is a difficult problem in itself. To overcome the above mentioned problem, Wu, Lin and Huang (2001) had applied Bayesian approaches but due to the high dimensionality problem with increasing number of parameters, this is

computationally more expensive. This excludes it from use in a real-time implementation.

Model based approaches are robust in gesture recognition, that is the reason why this approach is one of the most active approach for hand gesture recognition research (Stenger, Mendonca and Cipolla 2001; Wu, Lin and Huang 2001; Kim, Albuquerque, Havemann et al. 2004; Wang, Zhang and Dai 2007). However, model based approaches tend to suffer greatly from self-occlusion of the hand, variation in background color and most importantly they are computationally very expensive. This again poses problems for real-time implementation.

2.2. View Based Approach

The difficulty of parameter estimation in the model based approaches lead to the significant focus on the view based approaches, which are also known as appearance based approaches (Black and Jepson 1996; Cui and Weng 1996; Gupta, Mittal, Dutta et al 2002; Wong and Cipolla 2005). View based approaches use a single camera to capture the hand gesture. As the view based approaches only use the single camera, these methods are less complex, easy to implement, cost effective and computationally much faster than model based approaches. The key difference between model based and view based approaches is the feature extraction step. Model based approaches construct the 3D hand model from the multiple cameras view by applying stereo vision approach and extract the features by estimating the joint angle of fingers and palm; while view based approaches work by capturing 2D hand gesture images using single camera and there are various methods that exist for feature extraction. However, most of the view based approaches further apply the eigenvector approach to reduce the high dimension points of the image to low dimension points image. As eigenvector transfer or map high dimensions points to less dimensions points, which not only help to reduce the computational time dramatically but importantly also help to create the

user's hand shape, space and occlusion invariant solution. Finally, the extracted features are classified using machine learning algorithms.

Most of the early view based approaches used skin color as a key to find the motion regions (Stenger 2006). However, the skin color detection is highly effected by lighting conditions in the environment. An alternative new approach for features extraction is use of local scale-invariant features transform (SIFT) (Lowe 1999). Wang and Wang (2008) have shown encouraging results using SIFT. The problem with the SIFT is again the computationally expensive. The optic flow method used by Essa and Pentland (1997) is another view based method. However, optic flow method is more suitable to tack the overall motion of the body not the relatively small spatial area of the hand movement.

Another noble view based approach for feature extraction is called temporal template, a 2D image where the pixel value at each point represents the motion in that spatial location in the sequences of images, is introduced by Bobick and Davis (2001). The 2D vector image called Motion History Image (MHI) captures the spatial motion region on subsequent frames. MHI is further processed to find the motion gradient orientation (MGO) in each point and used as features vector for gesture classification (Bradski and Davis 2000). Due to the only 2D image calculation involve in MHI and MGO, the temporal template based method get advantage to run on real-time. Hence, this dissertation will explore the hand gesture recognition using the MHI and MGO approaches proposed by Bradski and Davis (2000) and Bobick and Davis (2001).

2.3. Gesture classification

After the features extraction, we could classify the features vector into a number of pre-define gestures. There are two major types of gesture classification method exist, which are defines as below.

2.3.1. Rule Based Method

In this approach, the manually encoded rules for gestures are compared with the input features vector. Rules are matched against the features vector and gesture associated with the best matching rule is declared as resultant gesture. For an example, Cutler and Turk (1998) used the rule based method to identify the actions based on hand motion. The effectiveness of this model is based on manual ability to encode rules for all gestures, which is the major limitation of this method for large sets of gestures. Also, there is a limit to the human ability to write rules for variation on hand shape, lighting changes and others.

2.3.2. Learning Based Method

The limitation of a human ability to find the relationship between high dimensions features sets and gestures encourages the alternative approach based on Machine Learning (ML) algorithms. As ML algorithms are better able to self learn the relationships between variables i.e. mapping between high dimensions features sets and gestures.

HMM (Starner, Weaver and Pentlan 1998; Lee and Kim 1999; Wilson and Bobick 1999; Marcel 2000; Nair and Clark 2000) is the most popular ML algorithm for gesture classification. The major reason for HMM's popularity is due to its success on similar kind of voice and hand writing recognition problems. The HMM algorithm is also not problem free. The major problem is the choice of the number of states and transitions. And more importantly the transitions from one stage to another stage, which is do not map well to the real-world processes (Derpanis 2004).

Alternatively, Time-Delay Neural Networks (TDNN) (Fels and Hinton 1997; Yang and Ahuja 1998), finite state machines (Bobick and Wilson 1997; Manresa, Varona, Mas et al 2005) and AdaBoost (Chen, Georganas and Petriu 2007; Francke, Ruiz-del-Solar and Verschae 2007) are also have been used for gesture classification problem, among them AdaBoost is

getting remarkably good success due to adaptive nature of its learning capability.

2.4. Summary of the Literature Review

In summary, though 3D model based approaches are robust but they are computationally expensive. Hence, this dissertation will focus on the view based approach. Extraction of the features vector though SIFT, Optic Flow and Haar-like (Lienhart and Maydt 2002; Chen, Georganas and Petriu 2007) methods have seen significant success but as this dissertation is mainly focused on low resource devices it will focus on the temporal template method. Use of the temporal template method requires low memory and processing capacity. We will experiment with NB classifier, RVM, RBF and MLP. However, we will also compare our results with Support Vector Machine (SVM), the de-facto standard of machine learning algorithms.

It is strongly believed that human vision system directly recognizes the movement from the motion itself i.e. without constructing the 3D model (Bobick and Davis 2001). This is another strong reason why we have chosen the vision based approach. Also, MHI represents the recency of the motion i.e. the *how* of the motion. Thresholding the MHI or MGO gives the region of the motion i.e. the *where* of the motion. *Where* or *how* is how we believe that human visual cortex process the visual information in the brain to recognize objects (Ungerleider and Mishkin 1982; Milner and Goodale 1995). That is another reason why it was decided to use the temporal template (i.e. MHI and MGO) based methods.

The main purpose of this dissertation is hand gesture recognition system for low resource devices. Hence, the focus will be on experiments with different algorithms and on comparisons of their advantages and disadvantages regarding their suitability for implementation on low resource devices rather than just focusing on improvement of one

classification algorithm's gesture classification rate. The methods used for experiments are described in detail in section 3.

3 Research Methodology

3. Research Methodology

Research methodology for real-time hand gesture recognition can be divided onto two parts. One is features extraction and another is gestures classification, both have been separately described below.

3.1. Feature Extraction

This dissertation uses the temporal template approach to extract the features vector. The features vector is created using Motion Gradient Orientation (MGO), which is proposed by Bradski and Davis (2000). Again, MGO is calculated using timed Motion History Image (tMHI), which is proposed by Davis (1999) and further improvement on Bobick and Davis (2001).

3.1.1 timed Motion History Image

To represent the *how* motion Bobick and Davis (2001) proposed the MHI. However, the representation of the MHI in floating point time manner i.e. timed motion history image (tMHI) is proposed by Bradski and Davis

(2000). tMHI is built by copying each frame's silhouette values with a floating point timestamp to the tMHI. The tMHI is represented as below,

$$tMHI\ (x,y) = \begin{cases} t & \textbf{\textit{if}}\ current\ silhoutte\ at\ (x,y) \\ 0 & \textbf{\textit{else if}}\ tMHI(x,y) < (t - \delta) \\ & \textbf{\textit{otherwise}}\ do\ nothing \end{cases} \qquad (3.1)$$

Where, t is current timestamp and δ is the maximum time duration to keep the motion history. The representation of the motion in timed manner makes the tMHI independent of the system speeds or frame rates within limits i.e. MHI capture the same area even with different frame rates (Bradski and Davis, 2000). Figure 3.1 shows a typical example of tMHI with a left moving hand, the tMHI value in other area than hand motions shows that the body movement is not controlled for while making the hand gestures.

Figure 3.1: tMHI example of move left hand gesture, where body movement has not been controlled.

3.1.2. Motion Gradient Orientation

If we take the gradient of the tMHI (ref. figure 3.1), we could get the direction of the motion, which will gives us a normal optical flow representation (Bradski and Davis 2000). The gradient of the tMHI could be easily calculated by convolving with Sobel filters in the *X* and *Y*

direction yielding the derivatives $F_x(x,y)$ and $F_y(x,y)$ respectively. Then, motion gradient orientation (MGO) at each pixel is given by,

$$\phi(x,y) = arctan\frac{F_y(x,y)}{F_x(x,y)} \qquad (3.2)$$

To avoid the gradient orientation on the edge of tMHI, which otherwise would negatively impact the MGO value and area of motion gradient, we initialize the MGO to zero where the neighboring differences are either too high (due to the larger temporal disparity) or too low (inside a silhouette). An example of MGO image for the tHMI show in figure 3.1 is shown in figure 3.2 below.

Figure 3.2: An example MGO image for move left hand gesture. The corresponding tMHI is show in figure 3.1.

3.1.3 Dimension Reduction

To create the features vector from the MGO image as shown in figure 3.2, 75% size of the MGO has been reduced i.e. reduced from 320x240 pixels size to 80x60 pixels by maintaining the height and width ratio. This needs to be done because otherwise it would require a high amount of memory for further processing and would be difficult to implement in low resource devices. Principle Component Analysis (PCA) has been used to generate the final features vector. The number of most variant or largest principle

components used was dependent upon the applied gesture classifier method. We have followed the following steps for PCA,

i. Input *training data* and number of largest or most variant principle components (*K*) to use for features extraction.
ii. Subtract the mean from *training data.*
iii. Calculate the covariance matrix of *training data*, *covResult* (using MATLAB's *cov* function).
iv. Calculate the eigenvector *V* (using MATLAB's eig function) using *covResult* matrix.
v. Desired *features vectors* = *training data* * *V* using *K* largest or most variant principle components.

The major reason for PCA is dimensions reduction, which not only helps to reduce the processing complexity but at the same time smoothes the noise up to a certain level.

3.2. Gesture Classification

The aim of this dissertation is to find the real-time gesture recognition methodology for low resource devices, hence we have experimented using different classification methodologies and also compare our results with most popular SVM classifier. Used classification methodologies are described below. Only 9 gestures are experimented with in this dissertation, hence all methods implemented 9 different small networks for each gesture.

3.2.1. Naïve Bayes

One of the simple Bayesian learning algorithms, which often performs better than complex algorithm in many cases, is called naïve Bayes

classifier. The basic principle behind naïve Bayes classifier is that it tries to maximize the posterior probability (MAP).

Suppose we would like to learn a function *f,* which would map a training sets *X* with attribute values *(x_1, x_2,….x_n)* to target *Y* with target possibilities (y_1, y_2,…y_n) i.e. *f: X->Y*. This problem can be formulate as MAP hypothesis as below,

$$y_{MAP} = \ argmax_{y_i \in Y} \ P(y_i | x_1, x_2, \dots x_n) \qquad (3.3)$$

We could rewrite equation (3.3) using Bayes theorem as below,

$$y_{MAP} = \underset{y_i \in Y}{argmax} \frac{P(x_1, x_2, \dots x_n \mid y_i)\, P(y_i)}{P(x_1, x_2, \dots x_n)}$$

As this is the maximization problem we could remove the denominator, hence,

$$y_{MAP} = argmax_{y_i \in Y} \ P(x_1, x_2, \dots x_n \mid y_i)\, P(y_i) \qquad (3.4)$$

As we assume that all attributes of *x* are independent to each other, hence we could rewrite equation 3.4 as below,

$$y_{MAP} = argmax_{y_i \in Y} \ P(y_i) \prod_{k=1}^{n} P(x_k | y_i) \qquad (3.5)$$

As features vectors are in continuous form, this dissertation follows common Gaussian approach by estimating mean and standard deviation for each combination of x_k and y_i to calculate the $P(x_k \mid y_i)$. The naïve Bayes classifier is very simple model to implement and it requires very little calculation. Hence, this is good candidate for gestures classification on low resource devices.

3.2.2. Sparse Bayesian Classifier

Sparse Bayesian classifier or Relevance Vector Machine (RVM) is a binary classifier, whose output is probabilistic. For N feature vectors or training sets x_n with target t_n, $\{x_n, t_n\}_{n=1}^{N}$, the classification problem is learning a function f so that features vector x_n will correctly map onto the correct class t_n. The probability of x_n to correctly classify t_n is given by $\sigma(y_n) = 1/(1 + e^{-y_n})$, where $y_n = f(x_n)$.

And the function f is define as,

$$f(x_n) = \sum_{m=1}^{M} w_m \emptyset_m(x_n) + w_0$$

Where, $M < N$, ω is weight, w_0 is bias term and $\Phi_m(x_m)$ is kernel function. This dissertation uses the Gaussian kernel with width 1.

The classification process of RVM use Bernoulli likelihood and sigmoid link function to calculate the $P(t|x)$. Hence, the likelihood is given by,

$$P(t|w) = \prod_{n=1}^{N} \sigma\{y(x_n; w)\}^{t_n} \; [1 - \sigma\{y(x_n; w)\}]^{1-t_n} \qquad (3.6)$$

Where, target $t_n \in \{0.1\}$ and ω is the weights vector. This process further follows the Laplace approximation procedure similar to MacKay (1992). The detail of the model is explained in Tipping (2001) and online adaptive training procedure in Tipping and Faul (2003). The major advantages of RVM are sparse solution i.e. a lower number of kernel points and probabilistic output. The sparse nature of the model means it requires less calculation and probabilistic output means we could in future supply this output as input to other systems if needed.

3.2.3. Multi-Layer Perceptron

Artificial Neural Networks (ANNs) are inspired by the observation that biological learning systems are build by complex webs of interconnected neurons (Mitchell 1997). Multi-layer perceptrons trained using *error back-propagation* algorithm are good for high dimensional non-linear area classification problem. A typical example of multi-layer perceptron with 2 layers and one hidden layer is shown below.

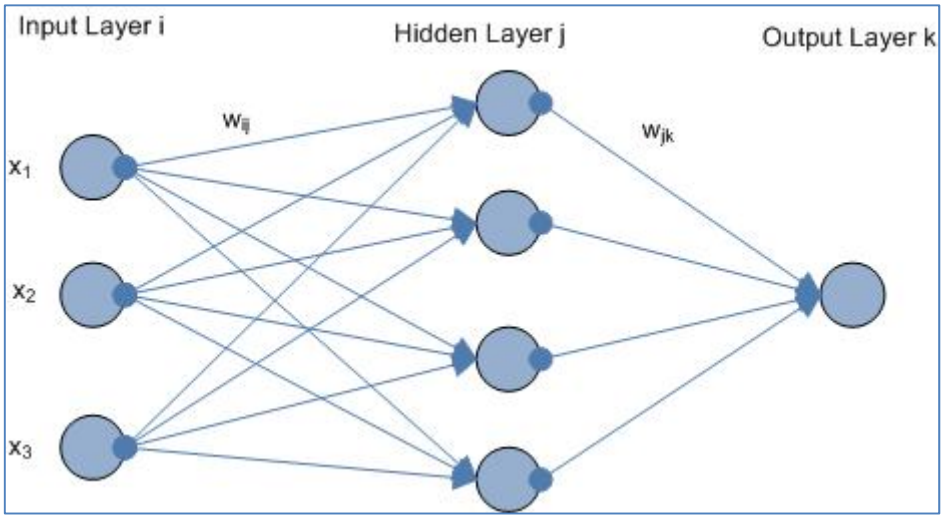

Figure 3.3: An example of 2-layers perceptron with one hidden layer.

The output of a node on ANNs is represented by,

$$x_j = \sum_{i=1}^{n} w_{ij}\, x_i + w_0 \qquad (3.7)$$

Where, x_j represent the output of the node, w_{ij} is the connection weight between input x_i node to output x_j node of forward layer and w_0 is the bias term.

The weight update rule using error back propagation algorithm is,

$$w_{ij} = w_{ij} + \eta * \Delta_j * x_i \qquad (3.8)$$

Where, w_{ij} is the connection weight between input x_i to $_{xj}$ node of forward layer and Δ_j is the summation of all errors caused by connection form j^{th} node to next layer's k nodes. However Δ_j is different for output layer and hidden layers,

For output layer,

$$\Delta_j = g'(in) * (Target - Output)$$

While for non output layers,

$$\Delta_j = g'(in) * \sum_k \Delta_k * w_{jk}$$

g' is derivative of the activation function g and 'in' represented the output of the node. This dissertation uses Mean Square Error (MSE) for error measurement. MLP with just 2 layers of weights are capable of approximating any continuous function (Bishop 1995), hence this dissertation will also experiment with MLP.

3.2.4. Radial Basis Function

Radial basis function (RBF) performs the exact interpolation of a set of points in a multi-dimensional space (Powell 1987).

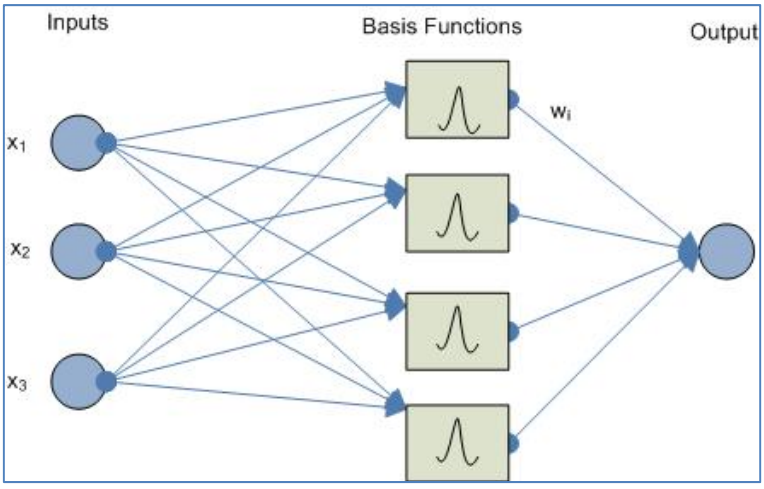

Figure 3.4: An example of RBF.

RBFs function are represented as below,

$$f(x) = w_0 + \sum_{j=1}^{K} w_j \Phi(\| x - x_j \|) \qquad (3.9)$$

Where, w_0 is the bias term, K is number of kernels, x_j is a kernel point, x is a data point and ϕ is kernel function. If there is N number of data then K should be less than or equal to N. This dissertation uses k-means algorithm to find the kernel points and the Guassian kernel function. The values of weights (w) are approximated using inverse of ϕ matrix and multiplying with targets of training samples. RBF is very fast to train, it also uses only a single hidden layer, which makes it less complex.

3.2.5. Support Vector Machine

Support Vector Machine (SVM) is the most popular classification algorithm of machine learning. It is developed by Vapnik (1979) and popularized by Schölkopf(1997), Vapnik (1998) and Burges (1998). SVM perform very well especially in high dimension data. As SVB first transfer the data from non-linearly separable data to linearly separable hyperplane then finds the classification boundary with equal distance from the both classes. More detail of the SVM can be found on Vapnik (1998). However, a brief description is defined here.

For the two class, c $\in$ {-1, 1}, classification problem using supervised learning method and the training set $\{x_i, c_i\}$ for $x_i \in R^n$, x is feature vector and R is n dimension hyperplane, there exist a following equation,

$$(w.x_i + b)c_i \geq 0 \qquad \forall i \qquad (3.10)$$

Where, w is the weight matrix and b is bias term.

4 Experiments and Results

4. Experiments and Results

The results presented below were taken as average of 25 repeats using 5-fold cross-validation. The used codes for these experiments were implemented using MATLAB 7.8 (R2009a) code and self written except RVM and SVM libraries. The experiments were executed on normal home use P4 personal computer with a 2.13GHz Intel® processor and 2GB of memory. A number of preset parameters were chosen so as to be suitable for low resource devices, the values of these preset parameters are separately described below along with the rationale behind their choice.

4.1 Description of Data

The experiments made use of 9 single hand gestures, they were: (1) bye, (2) come, (3) down, (4) go, (5) good luck, (6) left, (7) right, (8) up, and (9) victory. Sample tMHI and MGO for these gestures are shown in figure 4.1.

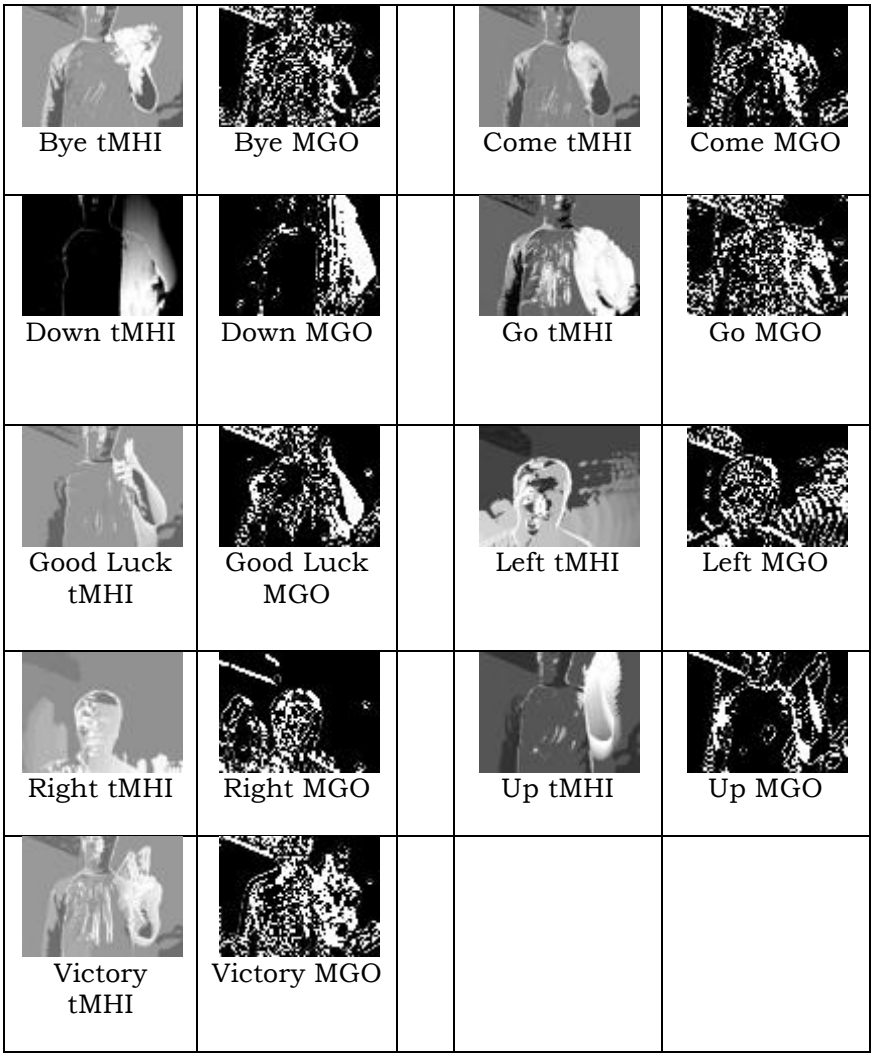

Figure 4.1: Example tMHI and MGO for the 9 gestures used in experiments.

The data was captured on unrestricted background and lighting condition using a laptop webcam at 320x240 pixels resolution and 15 frames per second. However, background objects were constant i.e. not moving. Four signers are used for experiments; two males and two females. Each signer was recorded 3 times for every gesture to give a total of 25 samples for each of the 9 gestures. The recording sessions for each signer were spread over 3 different times of day and under varying background and lighting conditions. The lighting conditions used were normal room lighting at day time, natural sunlight and neon tube lighting at night. The speed of the hand, body and head movements as well as the spatial location of the hand gestures was not controlled. The signers were instructed to allow

variations while making the gestures. This is clearly visible from observation of the tMHI and MGO images seen in figure 4.1.

In summary, four signers were used for data collection resulting in 9,500 hand gesture videos with approximately the same number for each gesture.

4.2 Assumption of Low Resource Device

In our context a low resource device means a system with a low resolution camera, small memory size, and slow processing capacity. The experiments were conducted on a relatively powerful desktop computer, therefore it was important that usage of memory and processing were restricted. This would allow easy transfer of the system to low resource devices. With this in mind two very crucial decisions were made regarding experiments. First, the MGO images were resized to 80x60 pixels before generating the features vector and second, the size of features vector (K) was restricted to be less than or equal to 30. Lower sizes of feature vector correspond to lower memory and computation resource requirements for the models; this made them more viable for low resource devices.

4.3 Parameters Tuning

Parameters tuning for any model always requires that very careful attention be paid to each variable. This is made more difficult when variables of the model are dependent to each other. Moreover, parameters tuning is very critical task when comparing the performance of different models as the performance of each model for a given problem is usually very sensitive to the parameters used. Hence, parameters tuning was done in two steps. As a first step the parameters for each model were tuned separately. After the parameters for each model had been tuned the

experiments were carried out to compare the models with each other. The size of the features vector had the single biggest influence in terms of performance and processing speed of the models.

4.3.1. Naïve Bayes

Naïve Bayes is easiest model to tune of those presented here. One important observation was when the number of negative training gestures increased the success of positive-positive tests would decrease relative to the success of negative-negatives tests, the inverse of this also held when the number negative training gestures were decreased. The reason for this was the higher prior probability for negative conditions due to higher quantity of negative training data. The numbers of positive and negative samples in the training data do not reflect the true operating conditions of the gesture recognition system. Hence, equal prior probability was used for all cases, which means only likelihood probabilities were able to influence the results.

The naïve Bayes classifier's performance was based on size of features vector (K) i.e. the number of features or number of most variant principal component used in PCA in this context. Experiments were conducted to find the best size for the features vector for the naïve Bayes classifiers.

Table 4.1 shows the success rate of hand gesture recognition for different value of K using the naïve Bayes classifier. It is clearly noticeable that for all 9 gestures, the success rate increases with the value of K. This is because as number of gestures increases the naïve Bayes model requires more features to optimally indentify the range of mean and standard deviation to associate them with a particular gesture. However, this does not mean that increase in K continuously increase the performance, as that will cause over fitting of the model after certain point. Proof of this is provide by the fact that the success rates was grater with values of K less than 30 when only two gestures were used (ref. table 4.1).

From the table 4.1 it is noticeable that left and right gestures are easier to recognize than up and down because they involve more movements then than up and down gestures, this can be seen in figure 4.1.

K	All gestures	Left and Right gestures	Up and Down gestures	Left and Right gestures by one signer	Up and Dow gestures by one signer
5	64	88	80	100	89
6	66	89	87	100	94
7	68	94	85	100	93
8	71	95	89	100	92
9	72	93	95	100	93
10	72	95	95	100	93
11	73	95	96	100	92
12	73	93	96	100	91
13	75	93	97	99	93
14	76	93	91	100	93
15	77	92	92	99	92
16	78	91	94	99	92
17	79	90	94	98	91
18	79	91	96	98	92
19	80	91	96	97	91
20	80	91	97	97	90
21	80	91	97	97	90
22	80	90	96	97	89
23	81	89	96	97	88
24	81	89	96	97	88
25	81	88	96	97	87
26	81	87	96	96	87
27	82	87	96	96	85
28	82	86	95	97	86
29	82	86	95	97	85
30	82	86	95	96	85

Table 4.1: Success rate of hand gesture recognition using naïve Bayes classifier for *K* equal to 5 to 30 have been listed for different combinations of gestures.

It might be interesting to look at how values of *K* greater than 30 would affect the success rates but this was not explored due to the time limitation for this research and the small devices assumption. For further experiments *K* equal to 27 was used for naïve Bayes classifier because this value gave the best results for all singers. However, *K* equal to 13 was used for single signer hand gestures recognition.

4.3.2 Sparse Bayesian Classifier

A library for sparse Bayesian classifier was used from Tipping (2009), which was based on Tipping (2001) and Tipping and Faul (2003). Different values for sparseness of the model and basis width were tried but the default values—70% for sparseness of the model and 0.05 for basis width—appeared to be optimal for the task i.e. there no significant improvement on RVM model with different values of the mentioned parameters.

K	All gestures	Left and Right gestures	Up and Down gestures	Left and Right gestures by single signer	Up and Dow gestures by single singer
5	50	49	48	46	48
6	50	49	48	46	48
7	50	49	48	46	48
8	50	49	48	46	48
9	50	49	48	46	48
10	50	49	48	46	48
11	50	49	48	46	48
12	50	49	48	46	48
13	50	49	48	46	48
14	50	49	48	46	48
15	50	49	48	46	48
16	50	49	48	46	48
17	50	49	48	46	48
18	50	49	48	46	48
19	50	49	48	46	48
20	50	49	48	46	48
21	50	49	48	46	48
22	50	49	48	46	48
23	50	49	48	46	48
24	50	49	48	46	48
25	50	49	48	46	48
26	50	49	48	46	48
27	50	49	48	46	48
28	50	49	48	46	48
29	50	49	48	46	48
30	50	49	48	46	48

Table 4.2: Success rate of hand gesture recognition using sparse Bayesian classifier for *K* equal to 5 to 30 have been listed for different combinations of gestures.

Table 4.2 shows the success rates for different value of *K* using RVM. It is clearly noticeable that *K* between 5 and 30 does not affect the performance of the RVM. More importantly, when all 9 gestures by all four signers were

used the success rate was slightly higher than when only 2 of the gestures or only a single signer were used. The success rate of RVM was lower than naïve Bayes, this has been discussed separately below. For further experiments with RVM a value of 5 was used for *K*.

4.3.3 Multi-Layer Perceptron

In this problem experiments show that a learning rate of 0.05 with momentum of 0.005 (10% of the learning rate) and approximately 400 iterations give the best results for MLP. However, it was observed that increases in the size of the features vector (i.e. number of input variables) increases the number of required iterations and causes the error rate to fluctuate. This is because the search space for optimum weights increases along with number of dimensions of the input features vector. This is because as the number of dimensions increases there is a larger area to search and more chances to get stuck in local optima.

Three layered MLP with one hidden layer was seen to give the best performance with no benefit being seen from additional hidden layers. The choice of the value of *K* and the number of hidden nodes was most difficult because both were highly dependent. For each value of *K* between 5 and 30, experiments were conducted with 3 to 30 hidden nodes. It would be unwise to present such a big table here so the results of the experiments will be briefly described instead. In all combinations of *K* and hidden nodes success rate was greater than 55. For all *K* with more than 6 hidden nodes success rate was greater than 65 and success rate was within the range of 65 to 73. Based on the experiments it was decided to use *K* equal to 8 and 12 hidden nodes. The success rate for the chosen combination was 72. Though the highest success rate of 73 was seen with 4 more hidden nodes it was decided to use the lower number based on the assumption of a low resource device.

4.3.4 Radial Basis Function

RBF has less numbers of parameters to tune than MLP. The most important parameter is the choice of kernel function, the number of kernels and their values. K-mean clustering was used to calculate the values for the kernels.

To take the decision on the value of *K* and number of kernels, for each value of *K* between 5 and 30 experiments were carried out with between 3 and 30 kernels. As before it would be unwise to present such a big table here. Hence only the results of the experiments will be briefly described here. The highest successes rate was 86 with *K* equal to 27 or more and the number kernels equal to 15 or more. Hence, for the further experiments with RBF, *K* equal to 27 and 15 kernels was used.

4.3.5 Support Vector Machine

MATLAB's SVM library had been used to test the SVM model. Over the linear, quadratic, polynomial and MLP, RBF as kernel function had performed the better in case of this dissertation.

Table 4.3 shows the success rate of hand gesture recognition for different values of *K* using SVM. It is clearly noticeable that effect of changes in *K* between 5 and 30 is significant. Importantly, the value of *K* between 5 and 8 performs better than higher values of *K*. As expected the recognition rate for single singers is better than when using all four signers. For further experiments *K* equal to 6 was used for SVM.

K	All gestures	Left and Right gestures	Up and Down gestures	Left and Right gestures by one signer	Up and Dow gestures by one signer
5	53	54	66	67	55
6	63	58	64	65	55
7	62	58	62	65	54
8	60	55	60	60	53
9	58	54	57	55	53
10	56	53	56	54	52
11	54	52	55	54	53
12	53	51	55	54	52
13	52	51	54	54	52
14	52	51	54	54	52
15	51	50	53	54	52
16	51	51	53	54	52
17	51	51	53	54	52
18	50	50	53	54	52
19	50	50	53	54	52
20	50	50	52	54	52
21	50	50	52	54	52
22	50	50	52	54	52
23	50	51	52	54	52
24	50	51	52	54	52
25	50	50	52	54	52
26	50	51	52	54	52
27	50	51	52	54	52
28	50	50	52	54	52
29	50	50	52	54	52
30	50	50	52	54	52

Table 4.3: Success rate of hand gesture recognition using sparse SVM for K equal to 5 to 30 have been listed for different combinations of gestures.

4.4 Results

In this section, the results of the experiments have been presented. Descriptions of the research environment and data have been provided on section 4.1. Implication of the experiments results have been described in section 5 separately.

4.4.1 Single Signer Hand Gestures Recognition

The success rates (as percentages) of the hand gestures recognition for a single signer are shown in table 4.4.

	RBF	MLP	NB	SVM	RVM
Bye	85	78	92	60	50
Come	98	86	94	79	50
Down	100	85	87	53	50
Go	87	72	91	66	50
Good Luck	97	76	94	53	50
Left	96	76	94	55	50
Right	100	95	100	56	50
Up	98	79	92	54	50
Victory	91	74	94	57	50
All gestures	**94.67**	**80.11**	**93.11**	**59.22**	**50**

Table 4.4: Success rates of single signer hand gestures recognition for different methods.

RBF outperforms the other methods for single signer hand gesture recognition. However, naïve Bayes classifier success rate of 93.11 on all gestures is only slightly less than the 94.67 success rate of the RBF. Moreover, NB outperforms RBF in cases of *bye*, *go* and *victory* gestures. RVM is not able to discriminate the single signer hand gestures as there are 50% positive and 50% negative test cases. SVM's overall success rate of 59.22 is not at all impressive even when compared to MLP, as MLP's overall success rate is 80.11.

The above experiments show that even the simplest algorithm (NB) could outperform the most advanced and complex algorithm (SVM). This clearly indicates that none of the algorithms are superior over others in all case—performance is problem specific.

4.4.2 Four Signers Hand Gestures Recognition

The success rates (in percentages) of the hand gestures recognition using data from all four signers is shown in table 4.5.

	RBF	MLP	NB	SVM	RVM
Bye	75	59	84	62	50
Come	81	73	80	77	50
Down	88	75	89	77	50
Go	79	61	73	64	50
Good Luck	89	72	81	62	50
Left	87	68	84	67	50
Right	95	76	88	81	50
Up	91	79	82	68	50
Victory	80	67	70	58	50
All gestures	**85.00**	**70.00**	**81.22**	**68.44**	**50**

Table 4.5: Success rates of four signers hand gestures recognition for different methods.

When data is used from all four signers, RBF and NB successfully classify 85% and 81.22% of the hand gestures respectively. However, differences in the success rate between NB and RBF is higher than in the single signer case. Again, NB outperforms RBF in cases of the *bye* and *down* gestures. RVM is still not able to discriminate the hand gestures when four signers are used as there are still 50% positive and 50% negative test cases. Importantly, SVM's overall success rate of 68.44 is higher than the 59.22 success rate in the single signer case, which clearly indicates that SVM performs better in higher dimension problems. Similar with RBF, MLP's success rate is decreased by approximately 10%.

4.4.3 Effect of Training Size

The size of training samples has crucial role in the implementation of all methods because in many cases collecting samples data is not only time consuming but is costly as well. Also, in supervised learning case such as this the labeling of training data is a tedious manual job. Hence the comparison of the success rate with various amounts of training data is important when comparing the different methods.

		RBF	MLP	NB	SVM	RVM
200 (100 positives and 100 negatives)	Bye	76	58	84	61	50
	Come	81	72	80	77	50
	Down	88	75	89	76	50
	Go	79	59	73	64	50
	Good Luck	88	73	82	62	50
	Left	87	68	81	67	50
	Right	95	75	90	81	50
	Up	90	80	80	68	50
	Victory	81	64	69	59	50
	All	**85.00**	**69.33**	**80.89**	**68.33**	**50**
100 (50 positives and 50 negatives)	Bye	72	57	83	65	50
	Come	79	74	77	69	50
	Down	87	71	86	69	50
	Go	76	57	67	57	50
	Good Luck	83	70	78	57	50
	Left	83	64	80	65	50
	Right	94	75	89	70	50
	Up	84	75	75	56	50
	Victory	75	64	73	63	50
	All	**81.44**	**67.44**	**78.67**	**63.44**	**50**
50 (25 positives and 25 negatives)	Bye	71	51	80	66	50
	Come	75	75	74	70	50
	Down	81	68	75	67	50
	Go	78	53	64	63	50
	Good Luck	80	70	75	59	50
	Left	75	57	84	65	50
	Right	95	73	79	75	50
	Up	74	72	77	50	50
	Victory	71	63	74	65	50
	All	**77.78**	**64.67**	**75.78**	**64.44**	**50**
30 (15 positives and 15 negatives)	Bye	61	60	75	57	50
	Come	58	73	74	73	50
	Down	71	66	67	70	50
	Go	81	51	53	60	50
	Good Luck	75	69	66	61	50
	Left	70	49	77	55	50
	Right	96	74	79	63	50
	Up	73	65	72	50	50
	Victory	71	53	71	56	50
	All	**72.89**	**62.22**	**70.44**	**60.56**	**50**

Table 4.6: Success rates of hand gestures recognition using different methods with 200, 100, 50 and 30 training sets for each gesture. There were 50% positive and 50% negative training samples in each case.

From table 4.6 it is clear that the success rates increase along with training data set size for each method except in the case of RVM. However,

RBF's success rate varies most while MLP's success rate varies less. When training size increased from 30 to 200 RBF's success rate increases by 12% and MLP's by 7% approximately. This clearly indicates that MLP could train using a smaller training set than RBF.

4.4.4 Training and Testing Duration

As this dissertation is aiming for real-time hand gestures recognition for small devices a comparison of processing time is essential. However, training time is also important because longer time always consume more resources.

	RBF	**MLP**	**NB**	**SVM**	**RVM**
450 training sets time (sec)	812.71	824.25	806.68	799.91	805.68
950 training sets time (sec)	1106.65	1114.66	1107.99	1106.63	1106.52

Table 4.7: Training time required for different methods in seconds, which also including 5-fold cross-validation.

The training time shown in table 4.7 includes MHI and MGO calculation, PCA for features extraction as well as 5-fold cross validation. From the experiment results shown in table 4.7 it is clear that the required training time for all methods is less than 20 minutes even with 950 hand gestures videos. The differences between the training times required for all methods are less than 30 seconds. From the observation of training time with 450 and 950 hand gestures, it is clear that training time is not linear with number of gestures used in training.

	RBF	MLP	NB	SVM	RVM
MHI update time (sec)	0.00370332	0.00370332	0.00370332	0.00370332	0.00370332
MGO calculation (sec)	0.0564	0.0564	0.0564	0.0564	0.0564
Features extraction	0.000129	0.000129	0.000129	0.000129	0.000129
1 gesture testing time (sec)	0.000039823	0.00000104292	0.00014070796	0.0000057522	0.000000428849
Total (sec)	0.060272143	0.06023336292	0.06037302796	0.0602380722	0.060232748849
15 frames per sec	**0.904082145**	**0.9035004438**	**0.9055954194**	**0.903571083**	**0.903491232735**

Table 4.8: Processing time required for different methods in seconds.

If the system is to run in real-time then processing time for gesture classification becomes much more important than training. As the system under discussion is a template based features extraction system it would be possible to continuously update a single tMHI. This would mean that to process each new video frame it would only be required to do the following jobs: (1) update the tHMI with new arrival frame, (2) calculate the MGO, (3) extract the features from the MGO using PCA and (4) classifying the gesture.

From the results of above experiments it can be seen that although the system could be implemented in real-time using a PC with a Core 2 Duo 2.13GHz processor it might not be feasible to implement in low resources devices as is. From the table 4.8, it is clearly noticeable that approximately 94% of the processing time is consumed by the MGO calculation. Hence we might look for an alternative to MGO calculation or at continuous MGO calculation in a similar way to tMHI. The reasons for the long processing time and how it could be improved in an implementation for low resource devices have been separately discussed in section 5.1.

5 Discussion and Conclusion

5. Discussion and Conclusion

5.1. Discussion

SVM is a state-of-the-art classifier, which outperforms the other methods in most of the problems and is useful in all kinds of problems and areas. RVM is another state-of-the-art method whose results are comparable with SVM. Moreover, the probabilistic output (posterior probability) and sparse solution of RVM makes it attractive over SVM. Therefore the highly uncompetitive results seen here for SVM and RVM when compared to RBF and NB is surprising and warrants further investigation on the implementation of SVM and RVM. For this dissertation the SVM library from MATLAB and RVM library from the Tipping (2009) were used. These implementations are considered reliable as evidenced by the similar results obtained by Wong and Cipolla (2005) in their experiment using the tMHI and MGO. However, Wong and Cipolla (2005) further improved the 50% success rate of hand gestures recognition using adaptive online

learning, which had not been done in this dissertation as it was not considered compatible with the goal of suitability for low resource devices.

SVM maps non-linear boundaries to linear ones then maximizes the decision boundary's distance with two groups. That is the reason why SVM works better in high dimensional problems. The video frames were reduced by 75%, which also reduced the dimensions of the features vector. This gives one reason why SVM was unable to perform as expected. For a given range of values for features (value of MGO in each pixels), SVM's kernels (also called support vectors) would be closer in lower dimension than they would be in higher dimensions, hence generally the distance of the boundary with different groups will be lesser, this causes poor generalization.

Though it is claimed that RVM usually gives results comparable to SVM in this case SVM correctly classified 68.44% of hand gestures while RVM managed only 50% (i.e. unable to discriminate for two class problem). As lesser number of samples and only four singers were used hence it might be the case that as RVM generally uses less kernels (sparse solution) it is less desirable in under sampled and hard to generalize problems (Chanel, Kierkels, Soleymani et al 2009). Another strong reason that spatial location of the hand gestures was not controlled and all signers had different physical characteristics (i.e. different size and shape of hands). It is also the case that the sample size was not large. This made estimation of the prior as well as likelihood probabilities poorer, which adversely affected the performance of the RVM.

Body movements, lighting conditions and movements of the background were not strictly controlled in these experiments. This is easily noticeable from 5 samples MGO images in Figure 5.1 for *bye* gesture using the same signer in the same location and recorded at the same time of day. RBF tends to perform better than MLP on noise data because RBF kernels do local approximations, where as MLP does global approximations, this is likely the reason why RBF performs better than MLP in this problem (Alejo, Garcia, Sotoca et el 2007).

Naïve Bayes is very simple to implement but one of the most effective algorithms of inductive learning. The performance of NB on hand gesture recognition is surprising here. It correctly classifies 81.22% of the hand gestures; this is approximately 13% more than SVM, one of the most complex and best ML algorithms. That the conditional independence assumption is rarely true in real time is major reason for its surprising performance, as local dependence of a node among each class and local dependencies of all nodes together, consistently or inconsistently play a major role. NB performs better when dependences are distributed evenly among classes or when dependencies cancel each other out (Zhang 2005).

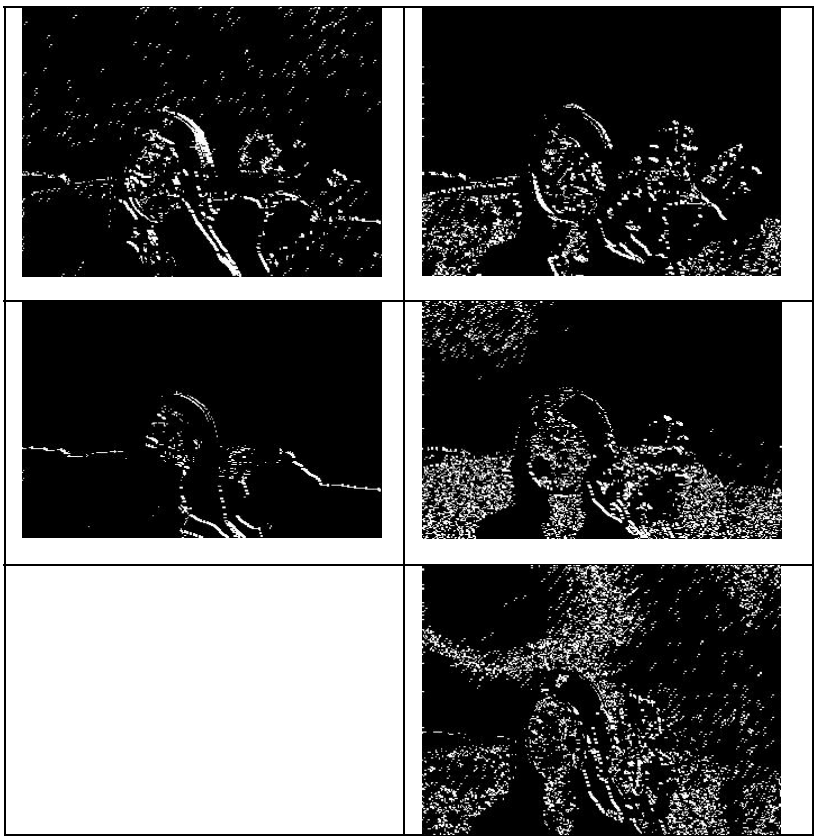

Figure 5.1: Sample MGO of Bye gestures by one singer

It is also worth mentioning here that for RBF, SVM, RVM and NB the optimum solution was found most of the time and that the error rate was consistent. In case of MLP however it was not able to find the optimum solution each time and the error rate was also not consistent, this was due

to the fact that the search space for the optimum solution increases with number of nodes in network as there are more and more chances to get stuck in local optima. The error rate for RBF, SVM and RVM also fluctuated with changing parameters values but NB use to highly consistence than other methods in this regard.

Although RBF and NB performed best they both required a much greater number of features than SVM. Also MLP require more features then SVM but still a lot less than RBF and NB. However, the increased performance given by greater numbers of feature for both RBF and NB dropped off after the features vector size passed 15. Hence, if the sensitivity of gestures recognition system is not very critical then the processing complexity of RBF and NB could be reduced by decreasing the size of features vector and sacrificing some accuracy in the classification rate.

The major objective of this dissertation is to develop a technique suitable for a real-time hand gestures recognition system for low resource devices but approximately 90% processing time required (ref. table 4.8; approximately 0.9 seconds processing time required each second) for all used methods even in personal computer is counterintuitive. It is easy to see from table 4.8 that MGO calculation alone consumes 93.5% of the required processing time. That is because the project was implemented in MATLAB which is not a suitable environment for low resource devices. As when similar method implemented by Wong and Cipolla (2005) in C++, the whole features extraction method took only 34.3 milliseconds, while in our MATLAB implementation only MGO extraction took 56.4 milliseconds. Hence, it is supposed that all the methods could be implemented more efficiently for use in real-time on low resource devices.

Hand gestures recognition is hot research topic with one and half decades of academic as well as industrial research. So far none of the systems has got a 100% success result in complex hand gestures recognition with a non-restricted environment. Does this mean we could never achieve 100% success rate? After going through all the experiments described here I strongly believe that we must need to look back on how biological vision

system works. Biological systems always outperform all artificial ones and are the best source for inspiration.

As pointed out by Wang, Zhang and Dai (2007), according to bionic view effective tracking of the object of interest is foremost important to the recognition of gestures, not the understanding of surrounding environment. Hence, I would like to argue that to build a robust hand gestures recognition system tracking of the hand is very important. When hand tracking is not used it is not possible to distinguish the movements of clothes, other body parts and background objects from the movement of the hand. Hence, no matter how much the features extraction and classification methods are improved we could not get robust results without good hand tracking.

One can argue that we can learn from the noise and that this is what AI aims for. Any algorithm can only learn if patterns or dependencies exist in the data. The hand gesture could be projected in any spatial location of the video frames, this means gestures as well as noise could be in any position in the video frame (i.e. there exists no pattern in the spatial location). Another possible method is to track hand colour but this is not a good solution as discussed in the literature review. In my opinion, tracking of the hand based on hand shape along with other cues is the best option for robust hand gestures recognition.

The argument above might suggest that 3D models are superior to view based hand gestures recognition models due to the fact that all 3D hand gestures recognition models first track the hand before representing it in 3D. However, I would like to make clear that I am only arguing for tracking of the hand i.e. tracking of object of interest in order to overcome noise. View based methods are important and a good first choice for real-time implementation due to processing advantage on 2D images over the greater complexity of processing in 3D models. For this reason view based approaches are rather important for low resources devices. Also,

experiments on lateral occipital complex (LOC[1]) by Kourtzi and Kanwisher (2001) suggest that shape of the object is the necessary condition to recognize an object, not the depth.

5.2. Applications of the System

Hand gesture recognition research is motivated by the many potential applications for human-machine interaction. Hand gestures recognition would be highly useful for mediating communication between hearing impaired and hearing people, instruction of home robots (Sing, Seth and Desai 2005), manipulation of virtual objects, computer games and other human-machine interactions. Also, many human behaviors are correlated with hand gestures, such as the clenched fist signaling confrontation, pointing with finger and others. Hence, it could be possible to extend this hand gestures methodology to human behavior analysis as well.

More importantly, this technology could be used for remote machine operation where direct human interaction involves high risk, such as land mine clearance, rescue, mining and other tasks. And another important application is in medical operations where it is important to avoid contamination by direct contact. Hence, diverse areas would see a benefit from hand gestures recognition technology.

[1] LOC: lateral occipital complex located on ventral visual pathway played major role on object recognition. On the experiment Kourtzi and Kanwisher (2001) found that LOC shows the neural adaptation when same shape objects are presented and not when different depth objects are show. Further neural adapts (stops or slow down firing) when inputs are same and starts firing when inputs are different.

5.3. Limitations of the Study

This research has number of limitations; some are due to the complexity of research problem and some are due to time limitation for this research. We have outlined major limitations as below.

1. In the samples collect the signer's body is slightly moving but background objects are constant. Hence, movement of background objects would affect the system adversely.
2. This research is limited to gesture recognition using one hand. This will limit the system in which gesture using two hands is required.
3. The experiment shown in section 4.4.3 shows that hand gesture recognition rate increases with the number of training set but it was not possible to test with larger training set as most of the hand gestures recognition research use to experiments with approximately more than three or four thousands hand gestures.
4. It would be nice to compare the hand gestures recognition rate of a signer, which was not included in training of the system. This has not been tested because we have neither enough hand gestures as mentioned in above limitation no. 3 nor was it possible to more signers. Without both more signers and more samples the experimental results would not have been meaningful.
5. Clothes of signer's are not moving in our samples data but in normal usage this could sometimes be the case, for example in windy environments.
6. The system was implemented in MATLAB which allowed quicker development times but gave reduced performance. If the system were implemented in C++ instead of in MATLAB it would almost certainly run faster.

5.4. Future Work

tMHI and MGO approaches looks very promising for hand gestures recognition. Though tMHI involves very low processing time, MGO calculation involves much more and consumes approximately 94% of the time required for the whole method (ref. section 4.4.4). Hence, research into continuous MGO update in place of the current approach of fully calculating it for each frame would be crucial for real-time implementation on low resource devices.

Another important area for future work would be to find a way of keeping repeated motion history on same spatial location without directly replacing with recent motion for tMHI as it is now. This would definitely help the tMHI and MGO method in those hand gestures where repetition of the hand movement is essential.

For the gesture recognition only the object of interest—the hand in this case—is important, neither the movement of the whole body nor the environment have any relevance. Hence, an effective hand tracking mechanism would be important for robust hand gesture recognition. This would allow only the motion of the hand to update the tMHI. Also, it would be interesting to see the community network of NB and RBF.

5.5. Conclusion

A hand gestures recognition problem using temporal template approach was experimented with; tMHI and MGO techniques were used to extract the features vector, which were then classified using five of the most popular machine learning algorithms. The experiments show encouraging results using RBF and NB, with RBF able to correctly classify 94.67% for single signer and 85% for multi-signers and NB able to correctly classify 93.11% for single signer and 81.22% for multi-signers. Hand gesture recognition rate was 70% for MLP and 68.44% for SVM, this was far lower

than the recognition rates of RBF and NB. A straight forward implementation RVM had a highly uncompetitive recognition rate.

The experiments suggest that RVM does not perform well with noisy data and SVM with low dimension data. Similarly MLP is also not a good classifier in noisy data due to its global boundary approximation approach. However, RBF could perform better even in noisy data as its local approximation of the boundary using kernel points provided an advantage, this is also another reason why RBF's performance improves with an increase in the number of kernels.

To implement any system in real-time, handling of noise (background objects moving, lighting conditions and others) is very important. Hence, I have argued that tracking of the hand is very important to develop the robust hand gestures recognition system, which is biologically as well as mathematically plausible.

Though the proposed system consumes the 90% of the resources in personal computer this could be easily reduced by 50% by implementing the system in C++ as evidenced by table 4.8 and Wong and Cipolla (2005). Hence, the proposed system could be implemented in low resource devices. However, for the real-time implementation, I would recommend the RBF method due to its high success rate and the fact that it could be trained using unsupervised method in addition to supervised methods.

In summary, even though we had not controlled the background, lighting condition and signer's body movement strictly, the success rates seen for RBF and NB are comparable to the best methods available. This dissertation shows that we could implement simple RBF and NB methods for real-time hand gestures recognition on low resource devices and still get results as good as the best of the currently available complex methods. The current correct classification rate of 85% using RBF and 81.22% using NB could be further improved by hand tracking to allow only the object of interest to be considered; there is strong evidence that this is what is done by the human vision system(Wang, Zhang and Dai 2007).

Bibliography

Alejo, R., Garcia, V., Sotoca, J.M., Mollineda, R.A. and Sánchez, J.S. (2007) Improving the Performance of the RBF Neural Networks Trained with Imbalanced Samples. *The 9th International Work-Conference on Artificial Neural Networks*, June 20-22, 2007, Spain, pp. 162-169.

Bauer, B., Kraiss, K.F. (2002) *Video-based sign recognition using self-organizing sub-units*. In: Proc. ICPR. 282–296

Binh, N.D., Shuichi, E. and Ejima, T. (2005). *Real-Time Hand Tracking and Gesture Recognition System*. In GVIP 05 Conference, 19-21 Dec 2005, CICC, Cairo, Egypt.

Bishop, C. M. (1995) Neural Networks for Pattern Recognition. *Oxford University Press*, UK.

Black, M. and Jepson, A. (1996) Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *In European Conference on Computer Vision,* pp. 329–342.

Bobick, A. F. and Davis, J.W. (2001) The recognition of human movement using temporal templates, *IEEE. Trans. on PAMI,* Vol. 23(3) pp. 257-267.

Bobick, A. and Wilson, A. (1997) A state-based approach to the representation and recognition of gesture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19(12) pp. 1325–1337.

Burges, C.J.C. (1998) A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, Vol. 2(2) pp. 121-167.

Bradski G. and Davis J. (2000) Motion Segmentation and Pose Recognition with Motion History Gradients. *IEEE Workshop on Applications of Computer Vision*, pp. 174-184.

Chanel, G., Kierkels, J.J.M., Soleymani, M., and Pun, T. (2009) Short-term emotion assessment in a recall paradigm. *International Journal of Human-Computer Studies*. Vol. 67 pp. 607-627

Chen, Q., Georganas, N.D. and Petriu, E.M. (2007) Real-time Vision based Hand Gesture Recognition Using Haar-like features. *IEEE Transactions on Instrumentation and Measurement.*

Cui, Y. and Weng, J. (1996) Hand sign recognition from intensity image sequence with complex backgrounds. *In IEEE Conference on Computer Vision and Pattern Recognition,* pp. 88–93.

Cutler, R. and Turk, M. (1998) View-based interpretation of real-time optical flow for gesture recognition. *In IEEE International Conference on Automatic Face and Gesture Recognition,* pp. 416–421.

Derpanis, G. K. (2004) A Review of Vision-Based hand Gestures.

Essa, I. and Pentland (1997), Coding, Analysis, Interpretation, and Recognition of Facial Expressions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 19 (7) pp. 757-763.

Fels, S. and Hinton, G. (1997) Glove-talk II: A neural network interface which maps gestures to parallel format speech synthesizer controls. *IEEE Transaction on Neural Networks*, Vol. 9(1) pp. 205–212.

Francke, H., Ruiz-del-Solar, J. and Verschae, R. (2007) Real-time Hand Gesture Detection and Recognition using Boosted Classifiers and Active Learning. pp. 533-547.

Garg, P., Aggarwal, N. and Sofat, S. (2009) Vision Based Hand Gesture Recognition. *In proceeding of world academy of science, engineering and technology* 37 January pp. 1024-1029.

Gupta, N., Mittal, P., Dutta Roy, S., Chaudhury, S. and Banerjee, S. (2002) Developing a gesture-based interface. *IETE Journal of Research*, Vol. 48(3) pp. 237–244.

Hassanpour, R., Wong, S. and Shahbahrami, A. (2008) Vision-Based Hand Gesture Recognition for Human Computer Interaction: A Review. *IADIS International Conference Interfaces and Human Computer Interaction.*

Kim, H., Albuquerque, G., Havemann, S. and Fellner, D.W. (2004) 3D Modeling with Hand Gesture Interaction in a Semi-Immersive Environment

Kourtzi, Z. and Kanwisher, N. (2001) Representation of preceived object shape by the human lateral occipital complex Science, Vol. 293 pp. 1506-1509. Avaliable from: www.web.mit.edu/bcs/nklab/media/pdfs/KourtziKanwisherScience01.pdf [accessed 6 March 2009]

Lee, H. and Kim, J. (1999) An HMM-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21(10) pp. 961–973.

Lienhart, R. and Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. In Proceeding *of IEEE International Conference Image Process*, Vol. 1 pp. 900–903.

Lowe, D. (1991). Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13(5) pp. 441–450.

Lowe, D. (1999). Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision,* Vol. 2 pp. 1150–1157.

MacKay, D.J.C. (1992). The evidence framework applied to classification networks. *Neural Computation*, Vol. 4(5) pp. 720-736.

Manresa, C., Varona, J., Mas, R. and Perales, F.J. (2005) Hand Tracking and Gesture Recognition for Human-Computer Interaction. *In Electronic Letters on Computer Vision and Image Analysis,* Vol. 5(3) pp. 96-104.

Marcel, S. (2000) Hand Gesture Recognition using Input Output Hidden Markov Models, *Proceeding of 4th IEEE International Conference on Automatic Face and Gesture Recognition,* pp. 456.

Milner, A.D. & Goodale, M.A. (1995) *The visual brain in action*. Oxford: Oxford University Press.

Mitchell, T.M. (1997) Machine Learning. McGraw-Hill Companies, Inc. International Edition.

Nair, V. and Clark J. (2002) Automated Visual Surveillance Using Hidden Markov Models, Vol. 02 pp. 88.

Pavlovic, V.I., Sharma, R. and Huang, T.S. (1997) Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19 pp. 677-695.

Powell, M.J.D. (1987) Radial basis functions for multivariable interpolation: A review. In Mason, J. C. and Cox, M. G. (Eds), Algorithms for Approximation, pp. 143-167. Oxford, Clarendon Press.

Rehg, J. and Kanade, T. (1994) Visual tracking of high DoF articulated structures: An application to human hand tracking. *In European Conference on Computer Vision,* pp. 35–46.

Scholkopf, B. (1997) Support vector learning. Oldenbourg, München, Germany [Note: Zugleich: Berlin, Techn. Univ., Diss., 1997], pp. 173.

Singh, R., Seth, B. and Desai, U.B. (2005) Vision based GUI for interactive mobile robots. *In proceeding of IU*, pp. 254-256.

Starner, T., Weaver, J. and Pentland, A. (1998) Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20(12) pp. 1371–1375.

Stenger, B. (2006) Template based Hand Pose recognition using multiple cues. *In Proceedings of 7th Asian Conference on Computer Vision*.

Stenger, B., Mendonca, P. and Cipolla, R. (2001) Model-based 3D tracking of an articulated hand. *In IEEE Conference on Computer Vision and Pattern Recognitio,* pp. 310–315.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research,* Vol. 1 pp. 211–244.

Tipping, M. E. and Faul, A.C. (2003) Fast marginal likelihood maximisation for sparse Bayesian models. In Bishop, C. M. and Frey, B. J. (Eds.), *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, Key West, FL, Jan 3-6.

Tipping, M.E. (2009) Sparse Bayesian Models (& the RVM) [online]. http://www.miketipping.com/index.php?page=rvm.

Ungerleider, L.G. and Mishkin, M. (1982) Two cortical visual systems. In D.J. Ingle, M.A. Goodale & R.J.W. Mansfield (Eds). *Analysis of Visual Behavior*. Cambridge, MA: MIT Press. pp. 549-586.

Vapnik, V. (1979). Estimation of Dependences Based on Empirical Data, Nauka, Moscow. (English translation: Springer Verlag, New York, 1982).

Vapnik, V. (1998) Stastistical learning theory. Wiley.

Vogler C. and Metaxas, D. (2001) A framework for recognizing the simultaneous aspects of American Sign Language. *Computer vision and image understanding*, Vol. 81 pp. 358–384

Wang, X., Zhang, X. and Dai, G. (2007) Tracking of Deformable Human Hand in Real Time as Continuous Input for Gesture-based Interaction. *Proceedings of the 12th international conference on Intelligent user interfaces*, pp 235 - 242

Wang, C.C. and Wang, K.C. (2008) Hand Posture recognition using Adaboost with SIFT for human robot interaction. *Springer Berlin*, ISSN 0170-8643, Vol. 370.

Wilson, A. and Bobick, A. (1999) Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21:9 pp. 884–900.

Wilson, A. and Bobick, A. (2000) Real-time online adaptive gesture recognition. *Proceeding of international conference on pattern recognition*. Vol. 1 pp. 270–275.

Wong, S. and Cipolla, R. (2005) Real-time adaptive hand motion recognition using a sparse Bayesian classifier. *Proceedings of the IEEE International Workshop on Human-Computer Interaction,* pp. 170-179.

Wu, Y. and Huang, T. S. (1999) Vision-Based Gesture Recognition: A Review. *In Gesture-Based Communication in Human-Computer Interaction*, Vol. 1739 pp. 103-115

Wu, Y., Lin, J. and Huang, T. (2001) Capturing natural hand articulation. *In IEEE International Conference on Computer Vision*, Vol. 2 pp. 426–432.

Yang, M. and Ahuja, N. (1998) Extraction and Classification of Visual Motion Patterns for Hand Gesture Recognition, *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 892

Zhang, H. (2005) The Optimality of Naive Bayes. Exploring conditions for the optimality of naive Bayes. In *International Journal of Pattern Recognition and Artificial Intelligence,* Vol. 19, No. 2.

Appendix A: Code

The code for this dissertation had written using MATLAB 7.8 (R2009a) version. Code fully utilizes the concept of object orientation methodology as well as modular programming concept suited for MATLAB programming. Each major function is written in separate file, so that we could reutilize those functions easily for later use. However, as code follows a kind of inheritance concepts, hence we could also simply call the hand gestures recognition using single command from the MATLAB command window. Also user could set all desire settings for hand gesture recognition using one single command called *SetUserOptions*, as well as using the *SetUserOptions.m* code file, hence they do not need to visit each function to changes the default parameters value.

The code for tMHI, MGO, PCA, features extraction, Naïve Bayes classifier, MLP, RBF and others are self-written by dissertation author. The code for this project has been optimized as well as cross tested many-times. MATLAB's SVM library has been used and RVM library has been used from the Tipping (2009).

All functions and variables name are written self-descriptive as much as possible. However, the purpose of the input and output parameters, variables and functions are separately written before beginning of each function and commented in code file itself.

All used code for this dissertation except SVM and RVM (Tipping 2009) have been listed below.

A.1 SetUserOptions.m

```
% function OPTIONS = SetUserOptions(varargin)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 Copyright University of Sussex
%
% Return the user options for hand gestures experiments.
%
% Input:
%   -varargin: Pairs of key and value to override the default values
%
% Output:
%   -OPTIONS: user options sturcture with data, parameters, and
experiment
%   models setting
%
% Dates:
%   -First Published: 1-Sept-2009

function OPTIONS = SetUserOptions(varargin)

    % Confirm that supplied arguments are must be in pairs
    if rem(nargin,2)
      error('Arguments to SetUserOptions should be in pairs of
property_name and value')
    end
    noNewSetting = nargin/2; %Number of new setting

    %Options about training
    OPTIONS.ExcludeTestingDataOnPCA = 0; %PCA Training and testing
data together or separately

    %Experiment with saved data in matlab file if LoadNewData = 0 else
load
    %using user setting
    OPTIONS.UseNosOfMostVariantAxes = 5; %Number of most variant axes
to use to generate training and testing data

    OPTIONS.RepeatExperiment = 25; %Number of time to repeat the
experiment
    OPTIONS.SaveResultsToFile = 0; %Save results of experiments to
file- 0 for default console and 1 for text file
    OPTIONS.SaveResultsFile = 'ResultFiles.txt'; %Results file name-
will be only use of the SaveResultsToFile = 1
    OPTIONS.LoadNewData = 0; %0 if read from saved file else 1 to read
from video files
    OPTIONS.SaveData = 1; %Save data to file so that later data
directly could be load without image processing
    OPTIONS.TestGestures = {'Bye', 'Come', 'Down', 'Go', 'Good Luck',
'Left', 'Right', 'Up', 'Victory'}; %Gestures to test
    %OPTIONS.TestGestures = {'Left', 'Right'};
    OPTIONS.ExperimentWithRBF = 1; %Flag value to whether RBF method
should be used or not for training and testing
    OPTIONS.ExperimentWithMLP = 1; %Flag value to whether MLP method
should be used or not for training and testing
```

```
    OPTIONS.ExperimentWithSVM = 1; %Flag value to whether SVM method
should be used or not for training and testing
    OPTIONS.ExperimentWithNB = 1; %Flag value to whether Naive Bayes
method should be used or not for training and testing
    OPTIONS.ExperimentWithRVM = 1; %Flag value to whether RVM method
should be used or not for training and testing

    %Options for root folder/location of the data
    OPTIONS.RootFolder = 'F:\Thesis\Data';
    %Options for the gestures folder for varities of data selection
    OPTIONS.Locations = {'Anita-Evening', 'Anita-Night','Rudra-
Day','Rudra-Evening','Rajendra-Evening', 'Pushmita-Night'};
%Sample/User folders
    %Number of defined gesture for supervised learning
    OPTIONS.Gestures = {'Bye', 'Come', 'Down', 'Go', 'Good Luck',
'Left', 'Right', 'Up', 'Victory'}; % type of gestures

    OPTIONS.DataFileName = 'Data_AnitaRudra'; %file name for data-
this will be saved in current location

    %Options related video processing to generate the Motion History
Images
    %and Motion Gradient Orientations
    OPTIONS.Delta = 2; %Time in Second, the duration for Motion
History Images
    OPTIONS.Frame_Buffer_Size = 4; %Buffer size to calculate the
frames difference
    OPTIONS.Use_AVIread = 1; %Option whether to read video using AVI
(1) or multimedia reader (0) library
    %Delta_Min/Max use to remove the noise from the Gradient
Orientation
    %Images
    OPTIONS.Delta_Min = 0.05; %Set the MGO to 0 if value of neighbour
is less than Delta_Min
    OPTIONS.Delta_Max = 0.5; %Set the MGO to 0 if value of neighbour
is greater than Delta_Max
    OPTIONS.Gradient_Epsilon = 0.00089; %Set the MGO to 0 if value of
X or Y gradient is less than Gradient_Epsilon

    %Options for the size of Input vector/data
    OPTIONS.MGO_Width = 200; %Resize frame's width size
    OPTIONS.MGO_Height = 200; %Resize frame's height size
    OPTIONS.MGOImages_Scale = 0.25; %Resize frame's size in scale by
maintaining aspect ratio

    %Options for research methodologies and error calculation
    OPTIONS.KFold = 5; %Number of K-Fold for generalization test- i.e.
K-Fold cross-validation

    %Options Radial Basis Function
    OPTIONS.RBF_K = 5; %Number of kernals for RBF

    %Options Multi-Layer Perceptron
    OPTIONS.Layers = [OPTIONS.UseNosOfMostVariantAxes 5 1]; %Number of
nodes per dimension- i.e. represent the numbers of layers as well as
number of nodes per layer
    OPTIONS.LearningRate = 0.05; %Learning rate
    OPTIONS.MomentumWeight = 0.005; %Momentum weight
    OPTIONS.MinimumMSE = 0.005; %Mean Square Error value to stop the
MLP training
```

```
    OPTIONS.Max_Epochs = 400; %Number of maximum iteration allowed for
MLP training

    %RVM parameter setting
    OPTIONS.Likelihood = 'Bernoulli'; %'Gaussian'
    OPTIONS.NoiseToSignal = 0.2;

    %Options to optimize the code
    OPTIONS.One_Eighty_By_PI = 57.2958; %180/PI - use One_Eighty_By_PI
instead of 180/PI many times

    %Overriding default value by user's paramenters
    for n=1:noNewSetting

        %Reading the name and value from the paired varargin
        propertyName    = varargin{(n-1)*2+1};
        newValue        = varargin{(n-1)*2+2};

        switch upper(propertyName)

            case 'EXCLUDETESTINGDATAONPCA'
                OPTIONS.ExcludeTestingDataOnPCA = newValue;
            case 'USENOSOFMOSTVARIANTAXES'
                OPTIONS.UseNosOfMostVariantAxes = newValue;
                OPTIONS.Layers = [OPTIONS.UseNosOfMostVariantAxes 9
1];
            case 'REPEATEXPERIMENT'
                OPTIONS.RepeatExperiment = newValue;
            case 'SAVERESULTSTOFILE'
                OPTIONS.SaveResultsToFile = newValue;
            case 'SAVERESULTSFILE'
                OPTIONS.SaveResultsFile = newValue;
            case 'LOADNEWDATA'
                OPTIONS.LoadNewData = newValue;
            case 'SAVEDATA'
                OPTIONS.SaveData = newValue;
            case 'TESTGESTURES'
                OPTIONS.TestGestures = newValue;
            case 'EXPERIMENTWITHRBF'
                OPTIONS.ExperimentWithRBF = newValue;
            case 'EXPERIMENTWITHMLP'
                OPTIONS.ExperimentWithMLP = newValue;
            case 'EXPERIMENTWITHSVM'
                OPTIONS.ExperimentWithSVM = newValue;
            case 'EXPERIMENTWITHNB'
                OPTIONS.ExperimentWithNB = newValue;
            case 'EXPERIMENTWITHRVM'
                OPTIONS.ExperimentWithRVM = newValue;
            case 'ROOTFOLDER'
                OPTIONS.RootFolder = newValue;
            case 'LOCATIONS'
                OPTIONS.Locations = newValue;
            case 'GESTURES'
                OPTIONS.Gestures = newValue;
            case 'DATAFILENAME'
                OPTIONS.DataFileName = newValue;
            case 'DELTA'
                OPTIONS.Delta = newValue;
            case 'FRAME_BUFFER_SIZE'
                OPTIONS.Frame_Buffer_Size = newValue;
```

```
                case 'USE_AVIREAD'
                    OPTIONS.Use_AVIread = newValue;
                case 'DELTA_MIN'
                    OPTIONS.Delta_Min = newValue;
                case 'DELTA_MAX'
                    OPTIONS.Delta_Max = newValue;
                case 'GRADIENT_EPSILON'
                    OPTIONS.Gradient_Epsilon = newValue;
                case 'MGO_WIDTH'
                    OPTIONS.MGO_Width = newValue;
                case 'MGO_HEIGHT'
                    OPTIONS.MGO_Height = newValue;
                case 'MGOIMAGES_SCALE'
                    OPTIONS.MGOImages_Scale = newValue;
                case 'KFOLD'
                    OPTIONS.KFold = newValue;
                case 'RBF_K'
                    OPTIONS.RBF_K = newValue;
                case 'LAYERS'
                    OPTIONS.Layers = newValue;
                case 'LEARNINGRATE'
                    OPTIONS.LearningRate = newValue;
                case 'MOMENTUMWEIGHT'
                    OPTIONS.MomentumWeight = newValue;
                case 'MINIMUMMSE'
                    OPTIONS.MinimumMSE = newValue;
                case 'MAX_EPOCHS'
                    OPTIONS.Max_Epochs = newValue;
                case 'LIKELIHOOD'
                    OPTIONS.Likelihood = newValue;
                case 'NOISETOSIGNAL'
                    OPTIONS.NoiseToSignal = newValue;
                case 'ONE_EIGHTY_BY_PI'
                    OPTIONS.One_Eighty_By_PI = newValue;
                otherwise,
                error('Unrecognised user option: ''%s''', propertyName)
            end%END: switch upper(propertyName)
        end%END: for n=1:noNewSetting

    end %END: function OPTIONS = SetUserOptions(varargin)
```

A.2 DataCapture.m

```
% function DataCapture(parentFolder)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 Copyright University of Sussex
%
% Use to capture the video training data for the system.
%
% Input:
%   -parentFolder: Parent folder to save the data
%
% Output:
%   -N/A
%
% Dates:
%   -First Published: 1-Sept-2009

function DataCapture(parentFolder)

    global fileIndex person

    fileIndex = 0;
    person = 'Pushmita';

    %Set parameters
    handGesture = {'Bye' 'Come' 'Down' 'Go' 'Good Luck' 'Left' 'Right'
'Up' 'Victory'}; % avialable gestures to record for for automatic data
labeling

    % Creating a video input object.
    %Input parameters are: device, device Id, format to view do
following
    %objinfo = imaqhwinfo('winvideo',1) then objinfo.SupportedFormats
    videoObjId = videoinput('winvideo',1,'YUY2_320x240'); %input
parameter should be system compatible

    % Creating a figure window and disabling all default behavior.
    hFigure = figure('Toolbar','none',...
       'Menubar', 'none',...
       'NumberTitle','Off',...
       'Name','Hand Motion- video recoding');

    % Create the image object in which we want to display the video
preview data.
    vidRes = get(videoObjId, 'VideoResolution');
    imWidth = vidRes(1);
    imHeight = vidRes(2);
    nBands = get(videoObjId, 'NumberOfBands');
    hImage = image( zeros(imHeight, imWidth, nBands) );

    % Creating list box for the selection of the guesture type
recoding
    hListBox = uicontrol('Style', 'listbox',...
       'String', 'Bye|Come|Down|Go|Good
Luck|Left|Right|Up|Victory',...
       'Position', [0 0 100 50]) ;
```

```
    % Creating list box for the selection of the guesture type
recoding
    hFileName = uicontrol('Style', 'edit',...
       'String', '',...
       'Position', [0 50 100 20]) ;

    % Creating Start Recoding push button
    uicontrol('String', 'Start Recording',...
        'Callback',
{@start_recoding,videoObjId,hImage,parentFolder,handGesture},...
        'Units','normalized',...
        'Position',[.18 0 0.15 .07]);

     % Creating Stop Recoding push button
    uicontrol('String', 'Stop Recoding',...
        'Callback', {@stop_recoding,videoObjId,hImage},...
        'Units','normalized',...
        'Position',[.34 0 0.15 .07]);

    % Creating start preview push button
    uicontrol('String', 'Start Preview',...
        'Callback', {@start_preview,videoObjId,hImage},...
        'Units','normalized',...
        'Position',[.51 0 0.15 .07]);

    % Creating stop preview push button
    uicontrol('String', 'Stop Preview',...
        'Callback', {@stop_preview,videoObjId},...
        'Units','normalized',...
        'Position',[.68 0 .15 .07]);

    % Creating close figure push button
    uicontrol('String', 'Close',...
        'Callback', {@close_program,videoObjId},...
        'Units','normalized',...
        'Position',[0.85 0 .15 .07]);

    % Specifing the size of the axes that contains the image object
    % so that it displays the image at the right resolution and
    % centers it in the figure window.
    figSize = get(hFigure,'Position');
    figWidth = figSize(3);
    figHeight = figSize(4);
    set(gca,'unit','pixels',...
            'position',[ ((figWidth - imWidth)/2)...
                         ((figHeight - imHeight)/2)...
                           imWidth imHeight ]);

    % Set up the update preview window function.
    setappdata(hImage,'UpdatePreviewWindowFcn',@frameUpdate);

    setappdata(hImage,'HandleOfListBox',hListBox);

    % Make handle to file name text control available to other
function.
    setappdata(hImage,'HandleOfFileName',hFileName);

end
```

```
function frameUpdate(obj,event,himage)
    % Example update preview window function.

    % Display image data.
    set(himage, 'CData', event.Data)

end

function start_preview(obj,event,vid,hImage)
    preview(vid, hImage);

    global fileIndex person

    fileIndex = fileIndex + 1;
    fileName = sprintf('%s%d',person,fileIndex);

    % Get handle of filename textbox control uicontrol.
    handleId = getappdata(hImage,'HandleOfFileName');
    % Get index of the selected text item on listbox
    set(handleId,'String',fileName);
end

function stop_preview(obj,event,vid)
    stoppreview(vid);
end

function start_recoding(obj,event,vid,hImage,parentFolder,handGesture)

    % Get handle of listbox control uicontrol.
    handleId = getappdata(hImage,'HandleOfListBox');
    % Get index of the selected text item on listbox
    gestureTypeId = get(handleId,'Value');

    % Get handle of filename textbox control uicontrol.
    handleId = getappdata(hImage,'HandleOfFileName');
    % Get index of the selected text item on listbox
    name = get(handleId,'String');

    gestureType = char(handGesture(gestureTypeId));
    filename =
sprintf('%s\\%s\\%s_%s.avi',parentFolder,gestureType,gestureType,name)
;

    %aviFileObject = avifile(filename, 'Colormap',gray(256));
    aviFileObject = avifile(filename);
    aviFileObject.Quality = 50;
    aviFileObject.Compression = 'None'; %'Indeo3' 'Indeo5' 'Cinepak'
'MSVC' 'None';

    vid.LoggingMode = 'disk&memory';
    vid.DiskLogger = aviFileObject;
    vid.TriggerRepeat = Inf;

    start(vid)

end
```

```
function stop_recoding(obj,event,vid,hImage)

    stop(vid)
    aviobj = close(vid.DiskLogger);
    clear aviobj
    %close(vid.DiskLogger);

    global fileIndex person

    fileIndex = fileIndex + 1;
    fileName = sprintf('%s%d',person,fileIndex);

    % Get handle of filename textbox control uicontrol.
    handleId = getappdata(hImage,'HandleOfFileName');
    % Get index of the selected text item on listbox
    set(handleId,'String',fileName);

end

function close_program(obj,event,vid)
    stoppreview(vid);

    delete(vid)
    clear vid

    close(gcf)
end
```

A.3 GestureRecognition.m

```
% function [success] GestureRecognition(ExecutionOption)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Main entry function for gesture recognition.
% It experiment according to the setting, eg. full experiment using k-
fold or
% view of Motion gradient Orientation etc
%
% Input:
%   -ExecutionOption: Option to execute the different method. 1- Test
MGO,
%   2- ExperimentGestureRecognition, 3-
ParamOptimizedGestureRecognition
%
% Output:
%   -success: return 1 if program execute successfully else 0
%
% Dates:
%   -First Published: 1-Sept-2009

function [success] = GestureRecognition(ExecutionOption)
    %Default value for success is 0
    success = 0;
    warning('off','stats:kmeans:EmptyCluster'); %Switch off off the
EmptyCluster warning

    %Blocks of code according to options
    if(ExecutionOption == 1)
        OPTIONS = SetUserOptions(); %Get user options setting
        TestMGO('N:\Thesis\Data\Rudra-Day\Bye', OPTIONS);


    elseif (ExecutionOption == 2) %ExperimentGestureRecognition

            OPTIONS = SetUserOptions('LoadNewData', 1 ...
                                    , 'DataFileName',
'Data_AnitaRudraPushmitaRajendra' ...
                                    , 'RootFolder', 'N:\Thesis\Data' ...
                                    , 'Locations', {'Anita-Evening',
'Anita-Night','Rudra-Day','Rudra-Evening','Rajendra-Evening',
'Pushmita-Night'} ...
                                    , 'Gestures', {'Bye', 'Come',
'Down', 'Go', 'Good Luck', 'Left', 'Right', 'Up', 'Victory'} ...
                                    , 'TestGestures', {'Bye', 'Come',
'Down', 'Go', 'Good Luck', 'Left', 'Right', 'Up', 'Victory'} ...
                                    , 'SaveResultsToFile', 1 ...
                                    , 'SaveResultsFile', 'Result.txt'
...
                                    , 'ExperimentWithRBF', 1 ...
                                    , 'ExperimentWithMLP', 1 ...
                                    , 'ExperimentWithSVM', 1 ...
                                    , 'ExperimentWithNB', 1 ...
                                    , 'ExperimentWithRVM', 1 ...
```

```
                                    , 'RepeatExperiment', 2 ...
                                    , 'UseNosOfMostVariantAxes', 5 ...
                                   ); %Get user options setting


        ExperimentGestureRecognition(OPTIONS);


    else %ParamOptimizedGestureRecognition

        UserMessage = 1;
        K_Start = 5;
        K_End = 6;

        fid = fopen('Rudra_RF.txt','wt'); % file to same the
optimization parameters

        OPTIONS = SetUserOptions('LoadNewData', 0 ...
                                    , 'DataFileName', '' ...
                                    , 'RootFolder', 'N:\Thesis\Data' ...
                                    , 'Locations', {'Rudra-Day','Rudra-
Evening'} ...
                                    , 'Gestures', {'Left', 'Right'} ...
                                    , 'TestGestures', {'Left', 'Right'}
...
                                    , 'SaveResultsToFile', 0 ...
                                    , 'SaveResultsFile', '' ...
                                    , 'ExperimentWithRBF', 1 ...
                                    , 'ExperimentWithMLP', 1 ...
                                    , 'ExperimentWithSVM', 1 ...
                                    , 'ExperimentWithNB', 1 ...
                                    , 'ExperimentWithRVM', 1 ...
                                    , 'RepeatExperiment', 25 ...
                                    , 'UseNosOfMostVariantAxes', -1 ...
                                   ); %Get user options setting
                                     %, 'ExcludeTestingDataOnPCA',1 ...


        if(UserMessage)
            fprintf(1,'\n Data Loading ... ');
        end
        [orginalData, group] = LoadData(OPTIONS);
        if(UserMessage)
            fprintf(1,'Finished. ');
        end

        noCols = size(orginalData,2);
        if(UserMessage)
            fprintf(1,'\n PCA ... ');
        end
        V = GetPrePCA_V_Matrix(orginalData);
        if(UserMessage)
            fprintf(1,'Finished. ');
        end

        for k=K_Start:K_End

            if(UserMessage)
                fprintf(1,'\n k = %d ... ',k);
```

```
            end

            %DBFile = sprintf('Data_%d',k);
            %OPTIONS.DataFileName =  DBFile;
            OPTIONS.UseNosOfMostVariantAxes =  k;
            OPTIONS.Layers = [OPTIONS.UseNosOfMostVariantAxes 9 1];

            data = orginalData * V(:,noCols-
OPTIONS.UseNosOfMostVariantAxes+1:noCols);

            ParamOptimizedGestureRecognition(OPTIONS, fid,data,group)

            %OPTIONS.LoadNewData =  0;
        end %END: for k=5:30
        fclose(fid);
        if(UserMessage)
            fprintf(1,'\n\n Finished.');
        end

    end

    warning('on','stats:kmeans:EmptyCluster'); %Switch on the
EmptyCluster warning back

    success = 1; % Return true when the function successfully executed

end %END: function [success] = GestureRecognition(ExecutionOption)
```

A.4 ExperimentGestureRecognition.m

```
% function ExperimentGestureRecognition(OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Experiment the gesture recognition using k-fold method for different
% methodologies and data according to user setting, have a look at
% SetUserOptions.m
%
% Input:
%   -OPTIONS - Options for the model to test
%
% Output:
%   -Empty

function ExperimentGestureRecognition(OPTIONS)

    %Check if need to save the result on the file
    if(OPTIONS.SaveResultsToFile)
        %Open the new file to save the experiments results
        %Format
        %Gesture, RBF, SVM, MLP, NB, RVM
        fid = fopen(OPTIONS.SaveResultsFile,'wt');

        fprintf(fid,'\t');
        if(OPTIONS.ExperimentWithRBF)
            fprintf(fid,'\tRBF');
        end
        if(OPTIONS.ExperimentWithSVM)
            fprintf(fid,'\tSVM');
        end
        if(OPTIONS.ExperimentWithMLP)
            fprintf(fid,'\tMLP');
        end
        if(OPTIONS.ExperimentWithNB)
            fprintf(fid,'\tNB');
        end
        if(OPTIONS.ExperimentWithRVM)
            fprintf(fid,'\tRVM');
        end

        fprintf(fid,'\n');
    end

    %START: Loop experiment for each gesture
    for iGesture=1:length(OPTIONS.TestGestures)
        %Load the data for experiment
        fprintf(1,'\nExperiment %s: load data. ',
OPTIONS.TestGestures{iGesture});
        [Model.Data.Train.Input Model.Data.Train.Group] =
GetData(OPTIONS.TestGestures{iGesture},OPTIONS);
        OPTIONS.LoadNewData = 0;

        if(OPTIONS.SaveResultsToFile)
            fprintf(fid,'%s\t',OPTIONS.TestGestures{iGesture});
```

```
        end

        %Experiment the data using RBF and print the error
        if(OPTIONS.ExperimentWithRBF)
            fprintf(1,' RBF.');
            errorRate = 0;
            for i=1:OPTIONS.RepeatExperiment
                errorRate = errorRate + ExperimentWithRBF(Model,
OPTIONS);
            end
            errorRate = errorRate/OPTIONS.RepeatExperiment;

            if(OPTIONS.SaveResultsToFile)
                fprintf(fid,'\t%3.2f',errorRate);
            else
                fprintf(1,' error=%3.2f ',errorRate);
            end
        end

        %Experiment the data using SVM and print the error
        if(OPTIONS.ExperimentWithSVM)
            fprintf(1,' SVM.');
            errorRate = 0;
            for i=1:OPTIONS.RepeatExperiment
                errorRate = errorRate + ExperimentWithSVM(Model,
OPTIONS);
            end
            errorRate = errorRate/OPTIONS.RepeatExperiment;

            if(OPTIONS.SaveResultsToFile)
                fprintf(fid,'\t%3.2f',errorRate);
            else
                fprintf(1,' error.=%3.2f ',errorRate);
            end
        end

        %Experiment the data using MLP and print the error
        if(OPTIONS.ExperimentWithMLP)
            fprintf(1,' MLP.');
            errorRate = 0;
            for i=1:OPTIONS.RepeatExperiment
                errorRate = errorRate + ExperimentWithMLP(Model,
OPTIONS);
            end
            errorRate = errorRate/OPTIONS.RepeatExperiment;

            if(OPTIONS.SaveResultsToFile)
                fprintf(fid,'\t%3.2f',errorRate);
            else
                fprintf(1,' error=%3.2f ',errorRate);
            end
        end

        %Experiment the data using Naive Bayes and print the error
        if(OPTIONS.ExperimentWithNB)
            fprintf(1,' NB.');
            errorRate = 0;
            for i=1:OPTIONS.RepeatExperiment
                errorRate = errorRate +
ExperimentWithNaiveBayes(Model, OPTIONS);
```

```
            end
            errorRate = errorRate/OPTIONS.RepeatExperiment;

            if(OPTIONS.SaveResultsToFile)
                fprintf(fid,'\t%3.2f',errorRate);
            else
                fprintf(1,' error=%3.2f ',errorRate);
            end
        end

        %Experiment the data using sparse Bayesian network and print
the error
        if(OPTIONS.ExperimentWithRVM)
            fprintf(1,' RVM.');
            errorRate = 0;
            for i=1:OPTIONS.RepeatExperiment
                errorRate = errorRate + ExperimentWithRVM(Model,
OPTIONS);
            end
            errorRate = errorRate/OPTIONS.RepeatExperiment;

            if(OPTIONS.SaveResultsToFile)
                fprintf(fid,'\t%3.2f',errorRate);
            else
                fprintf(1,' error=%3.2f ',errorRate);
            end
        end

        if(OPTIONS.SaveResultsToFile)
            fprintf(fid,'\n'); %Close the file handle
        end
    end
    %END: Loop experiment for each gesture

    if(OPTIONS.SaveResultsToFile)
        %Close the file handle and save the sive
        fclose(fid);
    end

end %END: ExperimentGestureRecognition(OPTIONS)
```

A.5 ParamOptimizedGestureRecognition.m

```
% function ParamOptimizedGestureRecognition(OPTIONS, fid,data,group)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Experiment for parameter optimization of gesture recognition models
according to user setting, have a look at
% SetUserOptions.m for detail
%
% Input:
%   -OPTIONS - Options for the model to test
%   -fid - file handle to write the results
%   -data - Data input for training and testing
%   -group - Label of <data> or class of <data>
%
% Output:
%   -Empty: However either results are use to save in file or print to
%   default console application based on the OPTIONS setting

function ParamOptimizedGestureRecognition(OPTIONS, fid,data,group)

    results = zeros(length(OPTIONS.TestGestures),30);
    %START: Loop experiment for each gesture
    for iGesture=1:length(OPTIONS.TestGestures)
        %Load the data for experiment
        [Model.Data.Train.Input Model.Data.Train.Group] =
GetTrainingData(OPTIONS.TestGestures{iGesture}, OPTIONS, data, group);

        %Experiment the data using RBF and print the error
        errorRate = 0;
        for i=1:OPTIONS.RepeatExperiment
            errorRate = errorRate + ExperimentWithRBF(Model, OPTIONS);
        end
        errorRate = errorRate/OPTIONS.RepeatExperiment;
        results(iGesture,1) = errorRate;


        %Experiment the data using SVM and print the error
        errorRate = 0;
        for i=1:OPTIONS.RepeatExperiment
            errorRate = errorRate + ExperimentWithSVM(Model, OPTIONS);
        end
        errorRate = errorRate/OPTIONS.RepeatExperiment;
        results(iGesture,2) = errorRate;


        %Experiment the data using MLP and print the error
        errorRate = 0;
        for i=1:OPTIONS.RepeatExperiment
            errorRate = errorRate + ExperimentWithMLP(Model, OPTIONS);
        end
        errorRate = errorRate/OPTIONS.RepeatExperiment;
        results(iGesture,3) = errorRate;
```

```
        %Experiment the data using Naive Bayes and print the error
        errorRate = 0;
        for i=1:OPTIONS.RepeatExperiment
            errorRate = errorRate + ExperimentWithNaiveBayes(Model,
OPTIONS);
        end
        errorRate = errorRate/OPTIONS.RepeatExperiment;
        results(iGesture,4) = errorRate;

        %Experiment the data using RVM and print the error
        errorRate = 0;
        for i=1:OPTIONS.RepeatExperiment
            errorRate = errorRate + ExperimentWithRVM(Model, OPTIONS);
        end
        errorRate = errorRate/OPTIONS.RepeatExperiment;
        results(iGesture,5) = errorRate;

        %Experiment the data using RBF and print the error
        for K=3:30

            OPTIONS.RBF_K = K; %Number of kernal for RBF

            errorRate = 0;
            for i=1:OPTIONS.RepeatExperiment
                errorRate = errorRate + ExperimentWithRBF(Model,
OPTIONS);
            end
            errorRate = errorRate/OPTIONS.RepeatExperiment;
            results(iGesture,K) = errorRate;
        end

        %Experiment the data using MLP and print the error
        for iHiddenLayer=3:30

            OPTIONS.Layers = [OPTIONS.UseNosOfMostVariantAxes
iHiddenLayer 1];

            errorRate = 0;
            for i=1:OPTIONS.RepeatExperiment
                errorRate = errorRate + ExperimentWithMLP(Model,
OPTIONS);
            end
            errorRate = errorRate/OPTIONS.RepeatExperiment;
            results(iGesture,iHiddenLayer) = errorRate;
        end

    end
    %END: Loop experiment for each gesture

    totalErrorRate = sum(results);
    totalErrorRate = totalErrorRate ./length(OPTIONS.TestGestures);

    fprintf(fid,'%d',OPTIONS.UseNosOfMostVariantAxes);
    for i=1:length(totalErrorRate)
        fprintf(fid,'\t%3.2f',totalErrorRate(i));
    end
    fprintf(fid,'\n');

end %END of function
```

A.6 ExperimentWithNaiveBayes.m

```
% function [cp] = ExperimentWithNaiveBayes(Model, OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Train and test the given MGO images using Gaussian Naive Bayes and
% return the error rate.
%
% Input:
%   -Model: Model which contains the data and other training info
%
%   -OPTIONS:  User options structure with Naive Bayes training
parameters info
%
% Output:
%   -ErrorRate: Error rate of K-Fold cross-validation
%
% Dates:
%   -First Published: 1-Sept-2009

function [ErrorRate] = ExperimentWithNaiveBayes(Model, OPTIONS)

    %Dividing data on K-Clusters for K-Fold training
    indicesKFold =
crossvalind('Kfold',Model.Data.Train.Group,OPTIONS.KFold);
    cp = classperf(Model.Data.Train.Group); %Initialize classperf
structure for error calculation

    %START: Loop for KFold test
    for i=1:OPTIONS.KFold
        test = (indicesKFold == i); train = ~test;
        if(OPTIONS.ExcludeTestingDataOnPCA)
            [trainingInput testingInput] =
PCA_TestSeparate(Model.Data.Train.Input(train,:),Model.Data.Train.Inpu
t(test,:),OPTIONS.UseNosOfMostVariantAxes);
        else
            trainingInput = Model.Data.Train.Input(train,:);
            testingInput = Model.Data.Train.Input(test,:);
        end
        %Train Input using Naive Bayes
        GNBModel = trainGaussianNaiveBayes(trainingInput,
Model.Data.Train.Group(train,:));
        %Predict Output using Naive Bayes Model
        classes = testGaussianNaiveBayes(GNBModel, testingInput);

        classperf(cp,classes,test); %Update the error rate using new
prediction

    end
    ErrorRate = cp.ErrorRate;

end


% function [GNBModel] = GaussianNaiveBayes(inputs,targets)
```

```
%
% Guassian Naive Bayes training routine.
%
% Input:
%   -inputs: Inputs data for training
%
%   -targets:  Targets class
%
% Output:
%   -GNBModel: Gaussian Naive Bayes Model which contains the prior
%   probabilities and means and standerd deviations per group per
features

function [GNBModel] = trainGaussianNaiveBayes(inputs, targets)

    UniqueClasses = sort(unique(targets)); %Find all unique classes
    NumberOfClasses = length(UniqueClasses); %Number of unique classes
    [NumberOfInputs NumberOfFeatures] = size(inputs); %Finding number
features and training sample size

    Means = zeros(NumberOfClasses,NumberOfFeatures); %Initialize Means
to hold mean values for unique combination of classes and features
    StandardDeviation  = zeros(NumberOfClasses,NumberOfFeatures);
%Initialize StandardDeviation to hold standard deviation values for
unique combination of classes and features
    PriorProbabilities = ones(NumberOfClasses,1); %Initialize
PriorProbabilities to hold Prior probability of each class
    %We assume that prior probabilities for each class is equal
    PriorProbabilities = PriorProbabilities .* (1/NumberOfClasses);
    %Or
    %PriorProbabilities = Number of each class/NumberOfInputs);

    %START: Loop through all classes
    for i=1:1:NumberOfClasses
        selectedClass = ( targets == UniqueClasses(i) ); %Selecting a
class at a time
        Means(i,:) = mean(inputs(selectedClass,:)); %Mean for selected
class, each feature wise
        StandardDeviation(i,:) = std(inputs(selectedClass,:));
%Standard Deviation for selected class, each feature wise
        %PriorProbabilities(i) = size(inputs(selectedClass,:),1);
    end
    %END: Loop through all classes
    %PriorProbabilities = PriorProbabilities ./size(inputs,1);

    %Zero standard deviation yield problem with logarithmic function
    %so change that to some default value
    stdZeros = (StandardDeviation == 0);
    StandardDeviation(stdZeros) = 0.15;

    %The the training values to model and return, which will be use
for
    %prediction
    GNBModel.UniqueClasses = UniqueClasses;
    GNBModel.NumberOfClasses = NumberOfClasses;
    GNBModel.NumberOfFeatures = NumberOfFeatures;
    GNBModel.PriorProbabilities = PriorProbabilities;
    GNBModel.Means = Means;
    GNBModel.StandardDeviation = StandardDeviation;
```

```
end

% function [results] = testGaussianNaiveBayes(GNBModel, inputs)
%
% Predict the output using trained Guassian Naive Bayes Model
%
% Input:
%   -GNBModel:  Gaussian Naive Bayes Trained model
%
%   -inputs: Inputs data for testing
%
% Output:
%   -Results: Prediction classes of the inputs

function [results] = testGaussianNaiveBayes(GNBModel, inputs)
    results = [];
    %START: Loop through all inputs
    for iInput=1:size(inputs)
        %Initialize as well as keep the logarithmic prior
probabilities
        %values for all classes
        probabilities = log(GNBModel.PriorProbabilities);
        %START: Loop through all classes
        for iClass=1:GNBModel.NumberOfClasses
            dblTemp = 0; %Hold the logarithmic sum of Gaussian
probability of class and feature
            %START: Loop through all features
            for iFeature=1:1:GNBModel.NumberOfFeatures
                %Logarithmic Gaussain Navie Bayes Formula for class
discrimination
                %if(DontAdd_2Pi)
                dblTemp = dblTemp - (inputs(iInput,iFeature)-
GNBModel.Means(iClass,iFeature))^2/(2*GNBModel.StandardDeviation(iClas
s,iFeature)^2)   - log(GNBModel.StandardDeviation(iClass,iFeature));
                %else
                %dblTemp=dblTemp-(inputs(iInput,iFeature)-
GNBModel.Means(iClass,iFeature))^2/(2*GNBModel.StandardDeviation(iClas
s,iFeature)^2)   -
log(sqrt(2*pi)*GNBModel.StandardDeviation(iClass,iFeature));
                %end
            end
            %END: Loop through all features
            probabilities(iClass) = probabilities(iClass) + dblTemp;
        end
        %END: Loop through all classes
        [value index] = max(probabilities); %Select the class with max
probabilities
        results = [results ; GNBModel.UniqueClasses(index)]; %Add up
the prediction output
    end
    %END: Loop through all inputs
end
```

A.7 ExperimentWithMLP.m

```
% function [cp] = ExperimentWithMLP(Model, OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Train and test the given MGO images using Multi-Layer Perceptron and
% return the error rate of K-fold cross-validation.
%
% Input:
%   -Model: Model which contains the data and other training info
%
%   -OPTIONS:  User options structure with MLP training parameters
info
%
% Output:
%   -ErrorRate: Error rate of K-Fold test
%
% Dates:
%   -First Published: 1-Sept-2009

function [ErrorRate] = ExperimentWithMLP(Model, OPTIONS)

    %Dividing data on K-Clusters for K-Fold training
    indicesKFold =
crossvalind('Kfold',Model.Data.Train.Group,OPTIONS.KFold);
    cp = classperf(Model.Data.Train.Group); %Initialize classperf
structure for error calculation

    %START: Loop for KFold test
    for i=1:OPTIONS.KFold
        test = (indicesKFold == i); train = ~test;
        if(OPTIONS.ExcludeTestingDataOnPCA)
            [trainingInput testingInput] =
PCA_TestSeparate(Model.Data.Train.Input(train,:),Model.Data.Train.Inpu
t(test,:),OPTIONS.UseNosOfMostVariantAxes);
        else
            trainingInput = Model.Data.Train.Input(train,:);
            testingInput = Model.Data.Train.Input(test,:);
        end
        %Train Input using MLP
        MLPModel =
trainMLP(trainingInput,Model.Data.Train.Group(train,:),OPTIONS.Learnin
gRate,OPTIONS.MomentumWeight,OPTIONS.MinimumMSE,OPTIONS.Layers,OPTIONS
.Max_Epochs);
        %Predict Output using MLP Model
        classes = classifyMLP(testingInput, OPTIONS.Layers,
MLPModel.weights);
        classperf(cp,classes,test); %Update the error rate using new
prediction

    end
    %END: Loop for KFold test
    ErrorRate = cp.ErrorRate;

end
```

```
% function MLPModel =
%
trainMLP(inputs,targets,learningRate,momentumWeight,minimumMSE,layers,
% maxEpochs)
%
% MLP training routine
%
% Input:
%   -inputs: Inputs data for training
%
%   -targets:  Targets class
%
%   -learningRate:  Learning rate for weights update
%
%   -momentumWeight:  Momentum weight to speed up training
%
%   -minimumMSE:  Mean Square Error to stop the training
%
%   -layers:  MLP structure which includes number of layers and
numbers of
%   node. Eg. 8 10 1, means 8 input nodes, 10 hidden nodes and 1
output
%   nodes
%
%   -maxEpochs:  Maximum epochs allowed for MLP training
%
% Output:
%   -MLPModel: MLP Model which contains the weights, training Mean
Square
%   Erro and Number of epoch used for training
%
% References: The MLP training code used form the Rudra PK Poudel,
Neural
% Network Assignment II

function MLPModel =
trainMLP(inputs,targets,learningRate,momentumWeight,minimumMSE,layers,
maxEpochs)

[trainingDataCount,inputNodesCount] = size(inputs);
[targetDataCount,targetNodesCount] = size(targets);

if trainingDataCount ~= targetDataCount

error('BackpropagationTraining:TrainingAndTargetDataLengthMismatch',
'The number of input vectors and desired ouput vectors do not match');
end

if length(layers) < 3
    error('BackpropagationTraining:InvalidNetworkStructure','The
network must have at least 3 layers');
end

if inputNodesCount ~= layers(1)
    msg = sprintf('Dimensions of input nodes (%d) does not match with
numbers of input layer (%d).',inputNodesCount,layers(1));
    error('BackpropagationTraining:InvalidInputLayerSize', msg);
end
```

```
if targetNodesCount ~= layers(end)
    msg = sprintf('Dimensions of output nodes (%d) does not match with
numbers of output layer (%d)',targetNodesCount,layers(end));
    error('BackpropagationTraining:InvalidOutLayerSize', msg);
end

leayersLength = length(layers);

%Initialize the weights matrix for MLP including bias nodes for all
layers
weights = cell(leayersLength-1,1);
for i=1:leayersLength-2
    weights{i} = [-1 + 2 .* rand(layers(i+1),layers(i)+1);
zeros(1,layers(i)+1)];
end
weights{end} = -2 + 2 .* rand(layers(end),layers(end-1)+1);

MSE = Inf; %Initialize default MSE to maximum
epochs = 0;

activation = cell(leayersLength,1);
activation{1} = [inputs ones(trainingDataCount,1)]; % activation{1} is
the input + 1 for the bias node activation
                                                    % activation{1}
remains the same throught the computation
for i=2:leayersLength-1
    activation{i} = ones(trainingDataCount,layers(i)+1); % inner
layers include a bias node (trainingDataCount-by-Nodes+1)
end
activation{end} = ones(trainingDataCount,layers(end)); % no bias node
at output layer

net = cell(leayersLength-1,1); % one net matrix for each layer
exclusive input
for i=1:leayersLength-2;
    net{i} = ones(trainingDataCount,layers(i+1)+1); % affix bias node
end
net{end} = ones(trainingDataCount,layers(end));

previousDeltaW = cell(leayersLength-1,1);
sumDeltaW = cell(leayersLength-1,1);
for i=1:leayersLength-1
    previousDeltaW{i} = zeros(size(weights{i})); % previousDeltaW
starts at 0
    sumDeltaW{i} = zeros(size(weights{i}));
end

% lowestsse = 999999;
% bestweights = weights;
while MSE > minimumMSE && epochs < maxEpochs

    for i=1:leayersLength-1
        net{i} = activation{i} * weights{i}'; % compute inputs to
current layer

        if i < leayersLength-1 % inner layers
            activation{i+1} = [1./(1+exp(-net{i}(:,1:end-1)))
ones(trainingDataCount,1)]; % for sigmoid
```

```
            %activation{i+1} = [(net{i}(:,1:end-1))
ones(trainingDataCount,1)]; %without sigmoid i.e for linear activation
function
        else % output layers
            activation{i+1} = 1 ./ (1 + exp(-net{i}));
            for iOutput=1:length(activation{i+1})
               if( activation{i+1}(iOutput)>=0.5)
                   activation{i+1}(iOutput)=1;
               else
                   activation{i+1}(iOutput)=0;
               end
            end
        end
    end

    % calculate sum squared error of all samples
    err = (targets-activation{end});       % save this for later
    sse = sum(sum(err.^2)); % sum of the error for all samples, and
all nodes
%     if(lowestsse>sse)
%         lowestsse = sse;
%         bestweights = weights;
%     end

    %delta = err .* activation{end} .* (1 - activation{end});
    delta = err;
    for i=leayersLength-1:-1:1
        sumDeltaW{i} = learningRate * delta' * activation{i};
        if i > 1
            delta = activation{i} .* (1-activation{i}) .*
(delta*weights{i});
            %delta =  (delta*weights{i}); % when there is no
activation
            %function
        end
    end

    % update the prev_w, weight matrices, epoch count and MSE
    for i=1:leayersLength-1
        previousDeltaW{i} = (sumDeltaW{i} ./ trainingDataCount) +
(momentumWeight * previousDeltaW{i});
        weights{i} = weights{i} + previousDeltaW{i};
    end
    epochs = epochs + 1;
    MSE = sse/(trainingDataCount*targetNodesCount);
    %MSEPerEpoch(epochs) = MSE;

end

% %printing error
% axis ( [0 length(MSEPerEpoch) 0 1]);
% x = 1:1:length(MSEPerEpoch);
% plot(x,MSEPerEpoch);
% s=input('Press enter to continue, enter 0 to stop \n');

% return the trained network
MLPModel.weights = weights;
% MLPModel.bestweights = bestweights;
MLPModel.epochs = epochs;
MLPModel.MSE = MSE;
```

```
end


% function Results = classifyMLP(inputs,layers, weights)
%
% Predict the output using trained MLP Model
%
% Input:
%   -inputs: Inputs data for training
%
%   -layers:  MLP structure which includes number of layers and
numbers of
%   node. Eg. 8 10 1, means 8 input nodes, 10 hidden nodes and 1
output
%   nodes. It should be same as used in training
%
%   -weights:  MLP trained weights
%
% Output:
%   -Results: Prediction of the inputs

function Results = classifyMLP(inputs,layers, weights)

[trainingDataCount,inputNodesCount] = size(inputs);

leayersLength = length(layers);

activation = cell(leayersLength,1);
activation{1} = [inputs ones(trainingDataCount,1)]; % activation{1} is
the input + 1 for the bias node activation
                                                    % activation{1}
remains the same throught the computation
for i=2:leayersLength-1
    activation{i} = ones(trainingDataCount,layers(i)+1); % inner
layers include a bias node (trainingDataCount-by-Nodes+1)
end
activation{end} = ones(trainingDataCount,layers(end)); % no bias node
at output layer

net = cell(leayersLength-1,1); % one net matrix for each layer
exclusive input
for i=1:leayersLength-2;
    net{i} = ones(trainingDataCount,layers(i+1)+1); % affix bias node
end
net{end} = ones(trainingDataCount,layers(end));

for i=1:leayersLength-1
    net{i} = activation{i} * weights{i}'; % compute inputs to current
layer

    if i < leayersLength-1 % inner layers
        activation{i+1} = [1./(1+exp(-net{i}(:,1:end-1)))
ones(trainingDataCount,1)]; % for sigmoid
        %activation{i+1} = [(net{i}(:,1:end-1))
ones(trainingDataCount,1)]; %without sigmoid i.e for linear activation
function
    else % output layers
        activation{i+1} = 1 ./ (1 + exp(-net{i}));
```

```
        Results = zeros(length(activation{i+1}),1);
        for iOutput=1:length(activation{i+1})
           if( activation{i+1}(iOutput)>=0.5)
               Results(iOutput)=1;
           else
               Results(iOutput)=0;
           end
        end
    end
end

%Results = Results';
end
```

A.8 ExperimentWithRBF.m

```
% function [cp] = ExperimentWithRBF(Model, OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Train and test the given MGO images using Radial Basis Function and
% return the rate using k-fold cross validation.
%
% Input:
%   -Model: Model which contains the data and other training info
%
%   -OPTIONS:  User options structure with RBF training parameters
info
%
% Output:
%   -ErrorRate: Error rate of K-Fold test
%
% Dates:
%   -First Published: 1-Sept-2009

function [ErrorRate] = ExperimentWithRBF(Model, OPTIONS)

    %Dividing data on K-Clusters for K-Fold training
    indicesKFold =
crossvalind('Kfold',Model.Data.Train.Group,OPTIONS.KFold);
    cp = classperf(Model.Data.Train.Group); %Initialize classperf
structure for error calculation

    %START: Loop for KFold test
    for i=1:OPTIONS.KFold
        test = (indicesKFold == i); train = ~test;
        if(OPTIONS.ExcludeTestingDataOnPCA)
            [trainingInput testingInput] =
PCA_TestSeparate(Model.Data.Train.Input(train,:),Model.Data.Train.Inpu
t(test,:),OPTIONS.UseNosOfMostVariantAxes);
        else
            trainingInput = Model.Data.Train.Input(train,:);
            testingInput = Model.Data.Train.Input(test,:);
        end
        %Train Input using RBF
        RBFModel =
trainRBF(trainingInput,Model.Data.Train.Group(train,:),OPTIONS.RBF_K);
        %Predict Output using RBF Model
        classes = classifyRBF(testingInput,RBFModel.C,
RBFModel.weights, RBFModel.sigmas);
        classperf(cp,classes,test); %Update the error rate using new
prediction

    end
    %END: Loop for KFold test

    ErrorRate = cp.ErrorRate;

end
```

```
% function RBFModel = trainRBF(inputs,targets, K)
%
% RBF training routine
%
% Input:
%   -inputs: Inputs data for training
%
%   -targets:  Targets class
%
%   -K:  Number of kernels
%
% Output:
%   -RBFModel: RBF Model which contains the weights, centers (C),
sigma and
%   K
%
% References: The RBF training code used form the Rudra PK Poudel,
Neural
% Network Assignment II

function RBFModel = trainRBF(inputs,targets, K)
    %Calculating Centroids for K-means
    exit = 0;
    while (exit~=1)
        AllowExit = 1;
        [IDX, C, sumd, D] = kmeans(inputs, K, 'emptyaction',
'singleton','replicates', 50);
        for i=1:size(C,1)
           for j=1:size(C,2)
               if(C(i,j)== 0)
                   AllowExit = 0;
               end
           end
        end
        if (AllowExit == 1)
            exit = 1;
        end
    end

    %Calculating the value of the sigma using distance between kernels
    sigmas = zeros(1,size(C,1));
    for i=1:size(C,1)
       max_distance = 0;
       for j=1:size(C,1)
          if( i~=j)
              distance = norm(C(i,:)-C(j,:));
              max_distance = max_distance + (distance^2);
              if(distance>max_distance)
                  max_distance = distance;
              end
          end
       end
       sigmas(i) = max_distance;
    end

    fai = zeros(size(inputs,1),K);
    % Calculate output matrix (fae)
    for i=1:size(inputs,1)
       for j=1:K
           fai(i,j) = nodeActivation(inputs(i,:), C(j,:), sigmas(j),
1);
```

```
        end
    end
    fai = [fai ones(size(fai,1),1)]; %adding bias
    %plot(1:size(fai,2),fai)

    try
        RBFModel.weights = pinv(fai) * targets;
        RBFModel.C = C;
        RBFModel.sigmas = sigmas;
    catch PINV_ERROR
        RBFModel = trainRBF(inputs,targets, K);
    end
end

% function output = nodeActivation (input, centre, sigma, option)
%
% Give the kernel output
%
% Input:
%   -inputs: A Input data
%
%   -centre:  A kernel
%
%   -sigma:  Value of sigma
%
%   -option:  Kernel function to be used
%
% Output:
%   -output: Given Kernel output for the given one point/input

function output = nodeActivation (input, centre, sigma, option)
    r = norm(input - centre);
    if (option == 1) %guassian
        output = exp(- (r^2)/(2*(sigma^2)) );
    elseif (option == 2) %Multiquadric
        output = sqrt( (r^2) + 0.3);
    elseif (option == 3) %Inverse multiquadrics
        output = 1 / (sqrt( (r^2) + 0.3));
    elseif (option == 4) %
        output = exp(- (r^2)) + 0.1;
    elseif (option == 5) %
        output = r;
    end

end

% function results = classifyRBF(inputs,C, weights, sigmas)
%
% Predict the output using trained RBF Model
%
% Input:
%   -inputs: Inputs data for training
%
%   -C:  Selected kernels/centers for RBF model
%
%   -weights:  RBF trained weights
%
%   -sigmas:  Value of sigma for each kernel
%
% Output:
```

```
%   -Results: Prediction of the inputs

function [results] = classifyRBF(inputs,C, weights, sigmas)
    results = zeros(size(inputs,1),1);
    K = size(C,1);
    for i=1:size(inputs,1)
        nodeOutput = weights(K+1); %bias
        %nodeOutput = 0;
        for j=1:K
            nodeOutput = nodeOutput + ( weights(j) *
nodeActivation(inputs(i,:), C(j,:), sigmas(j), 1) );
        end
        if ( nodeOutput>0.5)
            results(i) = 1;
        else
            results(i) = 0;
        end

    end

end
```

A.9 ExperimentWithRVM

```
% function [ErrorRate]= ExperimentWithRVM(Model, OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Train and test the given MGO images using sparse Bayesian classifier
and
% return the error rate using k-fold cross validation.
%
% This method use the library developed by Tipping (2009)
%
% Input:
%   -Model: Model which contains the data and other training info
%
%   -OPTIONS:  User options structure with sparse Bayesian classifier
training parameters info
%
% Output:
%   -ErrorRate: Error rate of K-Fold test
%
% Dates:
%   -First Published: 1-Sept-2009


% SPARSEBAYESDEMO  Simple demonstration of the SPARSEBAYES algorithm
%
%   SPARSEBAYESDEMO(LIKELIHOOD, DIMENSION, NOISETOSIGNAL)
%
% OUTPUT ARGUMENTS: None
%
% INPUT ARGUMENTS:
%
%   LIKELIHOOD      Text string, one of 'Gaussian' or 'Bernoulli'
%   DIMENSION       Integer, 1 or 2
%   NOISETOSIGNAL   An optional positive number to specify the
%                   noise-to-signal (standard deviation) fraction.
%                   (Optional: default value is 0.2).
%
% EXAMPLES:
%
%   SPARSEBAYESDEMO("Bernoulli",2)
%   SPARSEBAYESDEMO("Gaussian",1,0.5)
%
% NOTES:
%
% This program offers a simple demonstration of how to use the
% SPARSEBAYES (V2) Matlab software.
%
% Synthetic data is generated from an underlying linear model based
% on a set of "Gaussian" basis functions, with the generator being
% "sparse" such that 10% of potential weights are non-zero. Data may
be
% generated in an input space of one or two dimensions.
%
% This generator is then used either as the basis for real-valued data
with
```

```
% additive Gaussian noise (whose level may be varied), or for binary
% class-labelled data based on probabilities given by a sigmoid link
% function.
%
% The SPARSEBAYES algorithm is then run on the data, and results and
% diagnostic information are graphed.
%

%
% Copyright 2009, Vector Anomaly Ltd
%
% This file is part of the SPARSEBAYES library for Matlab (V2.0).
%
% SPARSEBAYES is free software; you can redistribute it and/or modify
it
% under the terms of the GNU General Public License as published by
the Free
% Software Foundation; either version 2 of the License, or (at your
option)
% any later version.
%
% SPARSEBAYES is distributed in the hope that it will be useful, but
WITHOUT
% ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
or
% FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public
License for
% more details.
%
% You should have received a copy of the GNU General Public License
along
% with SPARSEBAYES in the accompanying file "licence.txt"; if not,
write to
% the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor,
Boston,
% MA 02110-1301 USA
%
% Contact the author: m a i l [at] m i k e t i p p i n g . c o m
%

function [ErrorRate]= ExperimentWithRVM(Model, OPTIONS)

    %Dividing data on K-Clusters for K-Fold training
    indicesKFold =
crossvalind('Kfold',Model.Data.Train.Group,OPTIONS.KFold);
    cp = classperf(Model.Data.Train.Group); %Initialize classperf
structure for error calculation

    %START: Loop for KFold test
    for i=1:OPTIONS.KFold
        test = (indicesKFold == i); train = ~test;
        if(OPTIONS.ExcludeTestingDataOnPCA)
            [trainingInput testingInput] =
PCA_TestSeparate(Model.Data.Train.Input(train,:),Model.Data.Train.Inpu
t(test,:),OPTIONS.UseNosOfMostVariantAxes);
        else
            trainingInput = Model.Data.Train.Input(train,:);
            testingInput = Model.Data.Train.Input(test,:);
        end
        %Train Input using RVM
```

```
        RVMModel = trainRVM(trainingInput,
Model.Data.Train.Group(train,:),OPTIONS.Likelihood,OPTIONS.NoiseToSign
al);
        %Predict Output using RVM Model
        classes = testRVM(RVMModel, testingInput);
        classperf(cp,classes,test); %Update the error rate using new
prediction
    end
    %END: Loop for KFold test
    ErrorRate = cp.ErrorRate;

end

function[RVMModel] = trainRVM(inputs,groups, likelihood,
noiseToSignal)

    LIKELIHOOD  = SB2_Likelihoods(likelihood);
    [N  dimension] = size(inputs);  % Number of points
    basisWidth  = 0.05;     % NB: data is in [0,1]
    %
    % Define probability of a basis function NOT being used by the
generative
    % model. i.e. if pSparse=0.90, only 10% of basis functions (on
average) will
    % be used to synthesise the data.
    %
    pSparse     = 0.70;
    iterations  = 500;
    %
    % Heuristically adjust basis width to account for
    % distance scaling with dimension.
    %
    basisWidth  = basisWidth^(1/dimension);
    %


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    %
    % --- SYNTHETIC DATA GENERATION ---
    %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    %


    %
    % Now define the basis
    %
    % Locate basis functions at data points
    %
    C   = inputs;
    %
    % Compute ("Gaussian") basis (design) matrix
    %
    BASIS   = exp(-distSquared(inputs,C)/(basisWidth^2));
    %
    %
    % Randomise some weights, then make each weight sparse with
probability
```

```
    % pSparse
    %
    M           = size(BASIS,2);
    w           = randn(M,1)*100 / (M*(1-pSparse));
    sparse      = rand(M,1)<pSparse;
    w(sparse)   = 0;
    %
    % Now we have the basis and weights, compute linear model
    %
    z           = BASIS*w;
    %
    % Finally generate the data according to the likelihood model
    %
    switch (LIKELIHOOD.InUse)
     case LIKELIHOOD.Gaussian,
      % Generate our data by adding some noise on to the generative
function
      noise     = std(z) * noiseToSignal;
      Outputs   = z + noise*randn(N,1);
      %
     case LIKELIHOOD.Bernoulli,
      % Generate random [0,1] labels given by the log-odds 'z'
      Outputs   = groups; %double(rand(N,1)<SB2_Sigmoid(z));
    end
    %


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    %
    % --- SPARSE BAYES INFERENCE SECTION ---
    %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    %
    % The section of code below is the main section required to run
the
    % SPARSEBAYES algorithm.
    %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    %
    % Set up the options:
    %
    % - we set the diagnostics level to 2 (reasonable)
    % - we will monitor the progress every 10 iterations
    %
    OPTIONS     = SB2_UserOptions();
    %
    % Set initial parameter values:
    %
    % - this specification of the initial noise standard deviation is
not
    % necessary, but included here for illustration. If omitted,
SPARSEBAYES
    % will call SB2_PARAMETERSETTINGS itself to obtain an appropriate
default
    % for the noise (and other SETTINGS fields).
    %
```

```
    SETTINGS    = SB2_ParameterSettings('NoiseStd',0.1);
    %
    % Now run the main SPARSEBAYES function
    %
    [PARAMETER, HYPERPARAMETER, DIAGNOSTIC] = ...
        SparseBayes(likelihood, BASIS, Outputs, OPTIONS, SETTINGS);
    %
    % Manipulate the returned weights for convenience later
    %
    RVMModel.C = inputs(PARAMETER.Relevant,:);
    RVMModel.Weights = PARAMETER.Value;
    RVMModel.BasisWidth = basisWidth;
    RVMModel.LIKELIHOOD = LIKELIHOOD;


end

%*****************************************************

function [results] = testRVM(RVMModel, inputs)

    BASIS   = exp(-
distSquared(inputs,RVMModel.C)/(RVMModel.BasisWidth^2));
    results = BASIS * RVMModel.Weights;

    if(RVMModel.LIKELIHOOD.InUse == RVMModel.LIKELIHOOD.Bernoulli)
        toOne   = (SB2_Sigmoid(results)>0.5);
        results(toOne) = 1;
        results(~toOne) = 0;
    end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
% Support function to compute basis
%
function D2 = distSquared(X,Y)
    %
    nx  = size(X,1);
    ny  = size(Y,1);
    %
    D2 = (sum((X.^2), 2) * ones(1,ny)) + (ones(nx, 1) *
sum((Y.^2),2)') - ...
         2*X*Y';

end
```

A.10 ExperimentWithSVM.m

```
% function [cp] = ExperimentWithSVM(Model, OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Train and test the given MGO images using Support Vector Machine and
% return the error rate using K-fold cross validation.
%
% Input:
%   -Model: Model which contains the data and other training info
%
%   -OPTIONS:  User options structure with SVM training parameters
info
%
% Output:
%   -ErrorRate: Error rate of K-Fold test
%
% Dates:
%   -First Published: 1-Sept-2009

function [ErrorRate] = ExperimentWithSVM(Model, OPTIONS)
    %Dividing data on K-Clusters for K-Fold training
    indicesKFold =
crossvalind('Kfold',Model.Data.Train.Group,OPTIONS.KFold);
    cp = classperf(Model.Data.Train.Group); %Initialize classperf
structure for error calculation

    %START: Loop for KFold test
    for i=1:OPTIONS.KFold
        test = (indicesKFold == i); train = ~test;
        if(OPTIONS.ExcludeTestingDataOnPCA)
            [trainingInput testingInput] =
PCA_TestSeparate(Model.Data.Train.Input(train,:),Model.Data.Train.Inpu
t(test,:),OPTIONS.UseNosOfMostVariantAxes);
        else
            trainingInput = Model.Data.Train.Input(train,:);
            testingInput = Model.Data.Train.Input(test,:);
        end
        %Train Input using SVM
        %SVMStruct = svmtrain(trainingInput,
Model.Data.Train.Group(train,:));
        SVMStruct = svmtrain(trainingInput,
Model.Data.Train.Group(train,:),'Kernel_Function', 'rbf');
        %Predict Output using SVM Model
        classes = svmclassify(SVMStruct,testingInput);
        classperf(cp,classes,test);

    end
    %END: Loop for KFold test
    ErrorRate = cp.ErrorRate;

end
```

A.11 GetData.m

```
% function[data, group] = GetData(gesture, OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Return the training data and the class of the data for supervised
% learning. Data will be either load from the given file or processed
using
% gesture videos.
%
% Input:
%   -gesture: Data for experiment with one of the gesture from {'Bye',
%   'Come', 'Down', 'Go', 'Good Luck', 'Left', 'Right', 'Up',
'Victory'}
%
%   -OPTIONS: User options structure for locations/folders for data
%
% Output:
%   -data: Return the training data set
%
%   -group: gesture id as group for data/row feature vector
%
% Dates:
%   -First Published: 1-Sept-2009

function[data, group] = GetData(gesture, OPTIONS)

    %START: If OPTIONS.LoadNewData = 1
    if(OPTIONS.LoadNewData)

        [data, group] = LoadData(OPTIONS);

        if(OPTIONS.SaveData)
            %Saving the data for later use, which would save the time
to generate feature matrix
            SaveData(data,group,OPTIONS.DataFileName,0)
        end

        %PCA training input and select only given higest variant
principal
        %components
        [data] = PCA(data,OPTIONS.UseNosOfMostVariantAxes);

        if(OPTIONS.SaveData)
            %Saving data for later use, which would save the time
required for
            %feature calculation and PCA
             SaveData(data,group,OPTIONS.DataFileName,1)
        end

    end
    %END: If OPTIONS.LoadNewData

    if(OPTIONS.ExcludeTestingDataOnPCA)
        [data,group] = LoadDataFromFile(OPTIONS.DataFileName,0);
```

```
    else
        [data,group] = LoadDataFromFile(OPTIONS.DataFileName,1);
    end

    [data, group] = GetTrainingData(gesture, OPTIONS, data, group);

end

%Save the data according to the option for later use
function SaveData(data,group,fileName,pca)

    switch (fileName)

        case 'Data_AnitaRudra'
            if(pca)
                save Data_AnitaRudra_PCA data group
            else
                save Data_AnitaRudra data group
            end
        case 'Data_AnitaRudraPushmitaRajendra'
            if(pca)
                save Data_AnitaRudraPushmitaRajendra_PCA data group
            else
                save Data_AnitaRudraPushmitaRajendra data group
            end
        otherwise,
            error('Unrecognised file name: ''%s''', fileName)
    end
end

% Load the training data from the given file name for training
function [data,group] = LoadDataFromFile(fileName,pca)

    switch (fileName)

        case 'Data_AnitaRudra'
            if(pca)
                load Data_AnitaRudra_PCA data group
            else
                load Data_AnitaRudra data group
            end
        case 'Data_AnitaRudraPushmitaRajendra'
            if(pca)
                load Data_AnitaRudraPushmitaRajendra_PCA data group
            else
                load Data_AnitaRudraPushmitaRajendra data group
            end
        otherwise,
            error('Unrecognised file name: ''%s''', fileName)
    end
end
```

A.12 LoadData.m

```
% function[data, group] = LoadData(option)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% The function process the videos data from the OPTIONS setting
% and produce the inpute data for training
%
% Input:
%   -OPTIONS: User options structure for locations/folders for data
%
% Output:
%   -data: training dada processed from the gestures videos
%
%   -group: gesture id as group for data/row feature vector
%
% Dates:
%   -First Published: 1-Sept-2009

function[data, group] = LoadData(OPTIONS)

    data = [];
    group = [];

    %START: Loop over samples/users data
    for iLocations=1: length(OPTIONS.Locations)
       %START: Loop over gestures
       for iGesture=1:length(OPTIONS.Gestures)
           path =
sprintf('%s\\%s\\%s',OPTIONS.RootFolder,OPTIONS.Locations{iLocations},
OPTIONS.Gestures{iGesture});

           %Get the various images for a give location
           [MHIs, MGO] = GetMGOImages(path,OPTIONS);

           %Add one gesture video column wise i.e. number of column
equal to
           %number of gesture videos
           newData = GetMGOImagesMatrix(MGO,OPTIONS);
           data = [data; newData];
           group = [group; repmat(iGesture,[size(newData,1) 1])];

           clear MHIs MGO newData
           clear memory

       end %END: Loop over gestures
    end %END: Loop over samples/users data

    %Randomize the row/sample order
    newIndices = randperm(length(group));
    data = data(newIndices,:);
    group = group(newIndices,:);

end
```

A.13 GetFeatures.m

```
% function [data, group] = GetFeatures(locations,groupValue, OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% This function load the features of specified locations/folders
gesture videos
% using MHI and MGO methods
%
% Input:
%   -locations: locations/folders of the video
%
%   -groupValue: value for the group/class
%
%   -OPTIONS: User options structure for MHI, MGO and others
%
% Output:
%   -data: feature matrix a gesture per column (NOT PER ROW)
%
%   -group: value of group/class which is same as groupValue
%
% Dates:
%   -First Published: 1-Sept-2009

function [data, group] = GetFeatures(locations,groupValue, OPTIONS)
    numberOfFolder = size(locations,2);
    data = [];
    %START: Loop for all locations
    for i=1:numberOfFolder
        %Get the various images for a give location
        [MHIs, MotionGradientOrientations] =
GetMGOImages(locations{1},OPTIONS);

        %Add one gesture video column wise i.e. number of column equal
to
        %number of gesture videos
        data = [data;
GetMGOImagesMatrix(MotionGradientOrientations,OPTIONS)];

        clear MHIs ts masks MotionGradientOrientations mhiMasks
        clear memory

    end %END: Loop for all locations
    data = data';
    group = repmat(groupValue,[1 size(data,2)]); %Assign group value
for each gesture
end %END: function [data, group] = GetFeatures(locations,groupValue,
OPTIONS)
```

A.14 PCA.m

```
% function [pcaTrainData] = PCA(trainData,useNosOfMostVariantAxes)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% The function do the PCA for the given input data
%
% Input:
%   -trainData: training dada
%
%   -useNosOfMostVariantAxes: Number of most variant principle
component to
%   use to generate the project from the higher dimention to
%
%   -OPTIONS: User options structure for MHI, MGO and others
%
% Output:
%   -pcaTrainData: Output after the PCA
%
% Dates:
%   -First Published: 1-Sept-2009

function [pcaTrainData] = PCA(trainData,useNosOfMostVariantAxes)

    [noRows noCols] = size(trainData);

    meanValues = mean(trainData,1);
    trainData = trainData - repmat(meanValues,noRows,1);

    covResult = cov(trainData);
    %covResult(1:10,1:10)
    [V,D] = eig(covResult);
    %V(1:10,1:10)
    %D(1:10,1:10)
    %[V,D]

%     beforeSort = diag(D);
%     afterSort = sort(beforeSort);
%     diff = abs(beforeSort - afterSort);
%     sum(diff)
    pcaTrainData = trainData * V(:,noCols-
useNosOfMostVariantAxes+1:noCols);

    clear covResult V D noCols
end %END: [pcaTrainData] = PCA(trainData,useNosOfMostVariantAxes)
```

A.15 PCA_TestSeparate.m

```
% function [pcaTrainData,pcaTestData] =
% PCA_TestSeparate(trainData,testData,useNosOfMostVariantAxes)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% The function do the PCA for the given input data and testing data
will be
% projected based on the only training data- to make sure that
training set
% would not have any knowledge of testing data
%
% Input:
%   -trainData: Input training data
%
%   -testData: Input testing data
%
%   -useNosOfMostVariantAxes: Number of most variant principle
component to
%   use to generate the project from the higher dimention to
%
% Output:
%   -pcaTrainData: Output after the PCA
%
%   -pcaTestData: Output after the PCA of test data based on the
training
%   data
%
% Dates:
%   -First Published: 1-Sept-2009
function [pcaTrainData,pcaTestData] =
PCA_TestSeparate(trainData,testData,useNosOfMostVariantAxes)

    noCols = size(trainData,2);
    covResult = cov(trainData);
    [V,D] = eig(covResult);
    pcaTrainData = trainData * V(:,noCols-
useNosOfMostVariantAxes+1:noCols);
    pcaTestData = testData * V(:,noCols-
useNosOfMostVariantAxes+1:noCols);

    clear covResult V D noCols
end
```

A.16 GetMGOImages.m

```
% function [MHIs, MotionGradientOrientations]=
% GetMGOImages(location,OPTIONS)
%
% This function return the MHIs, Masks, MGO Images
% using MHI and MGO methods of the given location's videos
%
% Input:
%   -locations: locations/folders of the gesture video
%
%   -OPTIONS: User options structure for MHI, MGO and others
%
% Output:
%   -MHIs: Motion History Images
%
%   -MotionGradientOrientations: Motion Gradient Orientation images
for of
%   each MHI
%
% Dates:
%   -First Published: 1-Sept-2009

function [MHIs,MotionGradientOrientations]=
GetMGOImages(location,OPTIONS)

    list=dir(location);

    %if it read the system file names as well then remove those system
files, if they exist
    %they always use to be 2
    if list(1).name == '.',
      list = list(3:end);
    end
    N = size(list,1);

    %Allocate the memory
    MHIs = cell(N,1);
    MotionGradientOrientations = cell(N,1);

    %START: Loop through all gesture videos
    for i=1:N
        filename = strcat(location, '/',list(i).name);
        %Get the Motion History Image
        [MHIs{i}] =
GetMotionHistory(filename,OPTIONS.Delta,OPTIONS.Frame_Buffer_Size,OPTI
ONS.Use_AVIread);

        %Get the Motion Gradient Orientations image from the MHI
according
        %the use options
        [MotionGradientOrientations{i}] =
GetMotionGradientOrientations(MHIs{i}, OPTIONS.Delta_Min,
OPTIONS.Delta_Max,OPTIONS.Gradient_Epsilon,OPTIONS.One_Eighty_By_PI);
    end
    %END: Loop through all gesture videos
end
```

A.17 GetPrePCA_V_Matrix.m

```
% function [V] = GetPrePCA_V_Matrix(trainData)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% The function do the PCA for the given input data
%
% Input:
%   -trainData: training dada
%
%
% Output:
%   -V: Matrix V whose columns are the corresponding eigenvectors
%
% Dates:
%   -First Published: 1-Sept-2009

function [V] = GetPrePCA_V_Matrix(trainData)

    [noRows noCols] = size(trainData);

    meanValues = mean(trainData,1);
    trainData = trainData - repmat(meanValues,noRows,1);

    covResult = cov(trainData);
    %covResult(1:10,1:10)
    [V,D] = eig(covResult);

end %END: [V] = GetPrePCA_V_Matrix(trainData)
```

A.18 GetMGOImagesMatrix.m

```
% function [MGOMatrix] = GetMGOImagesMatrix(MGOImages,OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% This function generate the MGO Matrix from the given MGO images
%
% Input:
%   -MGOImages: MGO Images
%
%   -OPTIONS: User options structure for resize/rescale
%
% Output:
%   -MGOMatrix: MGO matrix i.e. feature matrix of given MGO Images
%
% Dates:
%   -First Published: 1-Sept-2009

function [MGOMatrix] = GetMGOImagesMatrix(MGOImages,OPTIONS)

    if(length(MGOImages)>0)
        aMGOImage = MGOImages{1};
        aMGOImage = imresize(aMGOImage,OPTIONS.MGOImages_Scale);
%resize the image
        [row col] = size(aMGOImage);
        MGOMatrix = zeros(length(MGOImages),row * col);
        for i=1:length(MGOImages);
            aMGOImage = MGOImages{i};
            %aMGO = imresize(aMGO,[NaN OPTIONS.MGO_Width]); %resize
the image
            aMGOImage = imresize(aMGOImage,OPTIONS.MGOImages_Scale);
%resize the image
            MGOMatrix(i,:) = aMGOImage(:); %convert image matrix to a
row vector
        end
        %MGOMatrix = MGOMatrix'; %convert rows vector to cols vector
    else
        MGOMatrix = []; %If size of MGOImages is 0 then return empty
matrix
    end
end %END: function [MGOMatrix] = GetMGOImagesMatrix(MGOImages,OPTIONS)
```

A.19 GraphPlot.m

```
% function GraphPlot()
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) Copyright University of Sussex
%
% Plot the graph for different hand gestures recognition model's
result.
%
% Input:
%   -None
%
% Output:
%   -Based on the called function. Either will write to file or
console
%   output
%
% Dates:
%   -First Published: 1-Sept-2009

function GraphPlot()

%     %Print the tables for NB
%     files = {'N:\Thesis\Working\Results\First with K-5 to
30\All_All.txt' ...
%         ,'N:\Thesis\Working\Results\First with K-5 to 30\All_LR.txt'
...
%         ,'N:\Thesis\Working\Results\First with K-5 to 30\All_UD.txt'
...
%         ,'N:\Thesis\Working\Results\First with K-5 to
30\Rudra_RF.txt' ...
%         ,'N:\Thesis\Working\Results\First with K-5 to
30\Rudra_UD.txt' ...
%         };
%
%
PrintSuccessRateTable(files,'RVM','N:\Thesis\Working\Results\First
with K-5 to 30\K_5to30_RVM.txt');

%     %Print the tables for All methods for single signer
%     files = {'N:\Thesis\Working\Results\Single
Signer\RBF_Single_Singer.txt' ...
%         ,'N:\Thesis\Working\Results\Single
Signer\MLP_Single_Singer.txt' ...
%         ,'N:\Thesis\Working\Results\Single
Signer\NB_Single_Singer.txt' ...
%         ,'N:\Thesis\Working\Results\Single
Signer\SVM_Single_Singer.txt' ...
%         ,'N:\Thesis\Working\Results\Single
Signer\RVM_Single_Singer.txt' ...
%         };
%
%
PrintSuccessRateTableAllGestures(files,'N:\Thesis\Working\Results\Sing
le Signer\Single_Singer.txt');

%     %Print the tables for All methods for single signer
```

```
%     files = {'N:\Thesis\Working\Results\All
Signers\RBF_All_Signers.txt' ...
%         ,'N:\Thesis\Working\Results\All Signers\MLP_All_Signers.txt'
...
%         ,'N:\Thesis\Working\Results\All Signers\NB_All_Signers.txt'
...
%         ,'N:\Thesis\Working\Results\All Signers\SVM_All_Signers.txt'
...
%         ,'N:\Thesis\Working\Results\All Signers\RVM_All_Signers.txt'
...
%         };
%
%
PrintSuccessRateTableAllGestures(files,'N:\Thesis\Working\Results\All
Signers\All_Singer.txt');

    %Print the tables for All methods for single signer
    files = {'N:\Thesis\Working\Results\Different Training
Size\30\RBF_All_Signers30.txt' ...
        ,'N:\Thesis\Working\Results\Different Training
Size\30\MLP_All_Signers30.txt' ...
        ,'N:\Thesis\Working\Results\Different Training
Size\30\NB_All_Signers30.txt' ...
        ,'N:\Thesis\Working\Results\Different Training
Size\30\SVM_All_Signers30.txt' ...
        ,'N:\Thesis\Working\Results\Different Training
Size\30\RVM_All_Signers30.txt' ...
        };


PrintSuccessRateTableAllGestures(files,'N:\Thesis\Working\Results\Diff
erent Training Size\30\All_Singer30.txt');

end

%Print the success rate on the given fileName by reading the results
from
%the files for the given model (successFor)
function PrintSuccessRateTable(files,successFor,fileName)

    result = [];
    K = [];
    for i=1:length(files)
        [K,RBF,SVM,MLP,NB,RVM] = textread(files{i},'%f %f %f %f %f
%f');
        if(strcmpi(successFor,'SVM'))
            result = [result SVM];
        elseif (strcmpi(successFor,'NB'))
            result = [result NB];
        elseif (strcmpi(successFor,'RVM'))
            result = [result RVM];
        end

    end
    result =  100 - (100 .* result);
    result = [K result];

    fid = fopen(fileName,'wt');
    [Rows,Cols] = size(result);
```

```
    for iRow=1:Rows
        for iCol=1:Cols
            if (iCol~=1)
                fprintf(fid,'\t');
            end
            fprintf(fid,'%2.0f',result(iRow,iCol));
        end
        if (iRow~=Rows)
            fprintf(fid,'\n');
        end
    end
    fclose(fid);

end

%Print the success rate on the given fileName by reading the results
from
%the files
function PrintSuccessRateTableAllGestures(files,fileName)

    result = [];
    for i=1:length(files)
        [errorRate] = textread(files{i},'%f');
        result = [result errorRate];
    end
    result =  100 - (100 .* result);

    avgSuccess = sum(result) ./ (size(result,1));

    fid = fopen(fileName,'wt');

    [Rows,Cols] = size(result);
    for iRow=1:Rows
        for iCol=1:Cols
            if (iCol~=1)
                fprintf(fid,'\t');
            end
            fprintf(fid,'%2.0f',result(iRow,iCol));
        end
        %if (iRow~=Rows)
            fprintf(fid,'\n');
        %end
    end

    [Rows,Cols] = size(avgSuccess);
    for iRow=1:Rows
        for iCol=1:Cols
            if (iCol~=1)
                fprintf(fid,'\t');
            end
            fprintf(fid,'%2.2f',avgSuccess(iRow,iCol));
        end
    end

    fclose(fid);

end
```

A.20 TestMGO.m

```
% function TestMGO(location, OPTIONS)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) 2009 University of Sussex
%
% Testing the MGO output to cross validate the features extraction
% functions. i.e. The main purpose of this file is to cross validate
the MHI and MGO.
%
% Input:
%   -location: location/folder of the videos
%   -OPTIONS: users options/model for experiment
%
% Output:
%   -Empty
%
% Dates:
%   -First Published: 1-Sept-2009

function TestMGO(location, OPTIONS)

    filename = 'N:\Thesis\Data\Rudra-Day\Victory\Victory_Rudra01.avi';
    %Get the Motion History Image
    [MHI] =
GetMotionHistory(filename,OPTIONS.Delta,OPTIONS.Frame_Buffer_Size,OPTI
ONS.Use_AVIread);

    %Get the Motion Gradient Orientations image from the MHI according
    %the use options
    [MGO] = GetMotionGradientOrientations(MHI, OPTIONS.Delta_Min,
OPTIONS.Delta_Max,OPTIONS.Gradient_Epsilon,OPTIONS.One_Eighty_By_PI);

    %Map delta duration to 0 to 255 to generate gray scale image
    %(gMHI)
    gMHI = MHI;
    tempUpdate = MHI>0;
    gMHI(tempUpdate)= MHI(tempUpdate) .* (255./OPTIONS.Delta);
    gMHI = uint8(  gMHI);

    gMHI = imresize(gMHI,OPTIONS.MGOImages_Scale); %resize the image
    MGO = imresize(MGO,OPTIONS.MGOImages_Scale); %resize the image

    imtool(gMHI)
    imtool(MGO)
    return;

    %Get MGO Images
    [MHIs, MotionGradientOrientations] =
GetMGOImages(location,OPTIONS);

    %Display the Images using imtool
    for i=1:5%length(MHIs)

        %Map delta duration to 0 to 255 to generate gray scale image
        %(mhiMasks)
```

```
        gMHI = MHIs{i};
        tempUpdate = MHIs{i}>0;
        gMHI(tempUpdate)= MHIs{i}(tempUpdate) .* (255./OPTIONS.Delta);
        gMHI = uint8(  gMHI);

        imtool(gMHI)

        %imtool(MotionGradientOrientations{i})
    end

%     data = GetMGOImagesMatrix(MotionGradientOrientations,OPTIONS);
%     [data] = PCA(data,OPTIONS.UseNosOfMostVariantAxes);
%     %[data] = PCA2(data',OPTIONS.UseNosOfMostVariantAxes);
%     size(data)

    clear MHIs ts masks MotionGradientOrientations mhiMasks
    clear memory
end %END: function TestMGO(location, OPTIONS)
```

A.21 GetTrainingData.m

```
% function[data, group] = GetTrainingData(gesture, OPTIONS, data,
group)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) Copyright University of Sussex
%
% Prepare the training and testing set from the input data based on
input
% gesture
%
% Input:
%   -gesture: Data for experiment with one of the gesture from {'Bye',
%   'Come', 'Down', 'Go', 'Good Luck', 'Left', 'Right', 'Up',
'Victory'}
%
%   -OPTIONS: User options structure for locations/folders for data
%
%   -data: input training data to generate the features vector
%
%   -group: gesture id as group for data/row feature vector
%
% Output:
%   -data: return the data
%
%   -group: gesture id as group for data/row feature vector
%
% Dates:
%   -First Published: 1-Sept-2009

function[data, group] =  GetTrainingData(gesture, OPTIONS, data,
group)

    if(~strcmpi(gesture,'All'))
        index = strmatch(gesture, OPTIONS.Gestures,'exact');
        positiveClass = group == index(1);
        positiveData = data(positiveClass,:);
        negativeData = data(~positiveClass,:);

        positiveCases = size(positiveData,1)
        negativeCases = size(negativeData,1);

        if(positiveCases>100)
            positiveData = positiveData(1:100,:);
        end
        positiveCases=100;

        if(negativeCases>positiveCases)
            negativeCases = positiveCases;
        end
        negativeData = negativeData(1:negativeCases,:);

        data = positiveData;
        group = repmat(1,[positiveCases 1]);

        data = [data; negativeData];
```

```
        group = [group; repmat(0,[negativeCases 1])];

        newIndices = randperm(length(group));
        data = data(newIndices,:);
        group = group(newIndices,:);
    end

end
```

A.22 GetMotionHistory.m

```
% function [MHI] =
% GetMotionHistory(filename,DELTA,FRAME_BUFFER_SIZE,USE_AVIREAD)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) Copyright University of Sussex
%
% Generate the Motion History image from the give video file.
%
% Input:
%   -filename: video's full path
%
%   -DELTA: Time in Second, the duration for Motion History Images
%
%   -FRAME_BUFFER_SIZE: Buffer size to calculate the frames difference
%
%   -USE_AVIREAD: ption whether to read video using AVI (1) or
multimedia
%   reader (0) library
%
% Output:
%   -MHI: MHI of the input video
%
% Dates:
%   -First Published: 1-Sept-2009

function [MHI] =
GetMotionHistory(filename,DELTA,FRAME_BUFFER_SIZE,USE_AVIREAD)

    %Variables initialization
    MHI = [,];
    previousFrames = cell(FRAME_BUFFER_SIZE,1);
    currentFrameIndex = 1;
    timestamp=0;
    timestampFactor = 0;

    %[pathstr, name, ext, ver] = fileparts(filename);
    if( USE_AVIREAD ) %if USE_AVIREAD = 1 (TRUE)

        %read the movie using matlab avi library
        mov = aviread(filename);
        numberOfFrames = size(mov,2);
        %START: Loop for each frame of the gesture video
        for i=1:numberOfFrames
           aFrame = frame2im(mov(i)); %converting a movie frame to
image
           if(i==1)
              try
                MHI = rgb2gray(aFrame); %converting frame image to
gray scale
              catch
                %Error will come if image is already in gray scale
                MHI = aFrame;
              end
              %allocating the first frame to index
              %2 to remove the all backgroud from the first frame only
              previousFrames{2} = MHI;
```

```
                MHI = MHI .* 0; %allocating the MHI size equal to frame
size
                for j=1:FRAME_BUFFER_SIZE
                    if(j~=2)
                      previousFrames{j} = MHI; %allocating all buffer
image size equal to frame size
                    end
                end
                MHI = double(MHI); %Changing data time of MHI to double

                %Calculating the time stamp factor i.e. real time
elapsed
                %between each frame
                movInfo = aviinfo(filename);
                timestampFactor = 1/movInfo.FramesPerSecond;
            end
            timestamp = timestampFactor*i; %total time elapsed

            %Updating MHI using new frame
            [MHI,previousFrames,currentFrameIndex] =
addNewFrameOnMHI(MHI, aFrame, previousFrames,
currentFrameIndex,timestamp,FRAME_BUFFER_SIZE,DELTA);
        end
        %END: Loop for each frame of the gesture video

        clear numberOfFrames

    else %if file format is non-avi
        mov = mmreader(filename);
        %START: Loop for each frame of the gesture video
        for i=1:mov.NumberOfFrames
            aFrame = read(mov,i); %reading ith movie frame
            if(i==1)
                try
                  MHI = rgb2gray(aFrame); %converting frame image to
gray scale
                catch
                  %Error will come if image is already in gray scale
                  MHI = aFrame;
                end
                 %allocating the first frame to index
                %2 to remove the all backgroud from the first frame only
                previousFrames{2} = MHI;
                MHI = MHI .* 0; %allocating the MHI size equal to frame
size
                for j=1:FRAME_BUFFER_SIZE
                    if(j~=2)
                      previousFrames{j} = MHI; %allocating all buffer
image size equal to frame size
                    end
                end
                MHI = double(MHI); %Changing data time of MHI to double

                %Calculating the time stamp factor i.e. real time
elapsed
                %between each frame
                movInfo = mmfileinfo(filename);
                timestampFactor = movInfo.Duration/mov.NumberOfFrames;
            end
            timestamp = timestampFactor*i; %total time elapsed
```

```
            %Updating MHI using new frame
            [MHI,previousFrames,currentFrameIndex] =
addNewFrameOnMHI(MHI,aFrame,previousFrames,currentFrameIndex,timestamp
,FRAME_BUFFER_SIZE,DELTA);
        end
        %END: Loop for each frame of the gesture video

    end

    %Adjust timestamp to 0 to DELTA
    keepMHIUpto = DELTA-timestamp;
    tempUpdate = MHI>0; %Adjust only non-zero fields
    MHI(tempUpdate) = MHI(tempUpdate) + keepMHIUpto;

    clear filename previousFrames currentFrameIndex timestampFactor
aFrame mov movInfo

end

% function [MHI,previousFrames,currentFrameIndex] =
%
addNewFrameOnMHI(MHI,aNewFrame,previousFrames,currentFrameIndex,timest
amp
% ,FRAME_BUFFER_SIZE,DELTA)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) Copyright University of Sussex
%
% Generate the Motion History image from the give video file.
%
% Input:
%   -MHI: MHI which will be update using current frame (aNewFrames)
%
%   -aNewFrame: New Frame to update MHI
%
%   -previousFrames: Previous frames
%
%   -currentFrameIndex: Index to update the buffer using current frame
%
%   -timestamp: Time stamp i.e. time elapsed from the starting of the
video
%
%   -FRAME_BUFFER_SIZE: the size of the buffer
%
%   -DELTA: Time duration used for MHI
%
% Output:
%   -MHI: Updated MHI
%
%   -previousFrames: Updated frame bufffer using current aNewFrames
%
%   -currentFrameIndex: Next buffer index to update
%    i.e. currentFrameIndex + 1, in cyclic order

function [MHI,previousFrames,currentFrameIndex] =
addNewFrameOnMHI(MHI,aNewFrame,previousFrames,currentFrameIndex,timest
amp,FRAME_BUFFER_SIZE,DELTA)
```

```
    %Converting the frame to gray scale and saving to buffer
    try
        previousFrames{currentFrameIndex} = rgb2gray(aNewFrame);
    catch
       %Error will come if image is already in gray scale
       previousFrames{currentFrameIndex} = aNewFrame;
    end

    %Calculating next frame index in cyclic order
    nextFrameIndex = currentFrameIndex+1;
    if(nextFrameIndex>FRAME_BUFFER_SIZE)
        nextFrameIndex = 1;
    end

    %Generate the silhouette using the frame difference then binary
thresh
    %holding
    silhouette = imabsdiff(previousFrames{currentFrameIndex},
previousFrames{nextFrameIndex});
    thresh = graythresh(silhouette);
    silhouette = (silhouette >= thresh * 255);

    %Updating the MHI using silhouette
    [MHI] = UpdateMotionHistory(MHI, silhouette, timestamp, DELTA);
    currentFrameIndex = nextFrameIndex;

    clear silhouette thresh nextFrameIndex

end
```

A.23 UpdateMotionHistory.m

```
% function [MHI] = UpdateMotionHistory(MHI, silhouette, timestamp,
DELTA)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) Copyright University of Sussex
%
% Update Motion History Image with the given silhoutte.
%
% Input:
%   -MHI: MHI which will be update using current silhouette
%
%   -silhouette: Silhouette to update MHI
%
%   -timestamp: Time stamp i.e. time elapsed from the starting of the
video
%    which will be use for MHI
%
%   -DELTA: Time in Second, the duration for Motion History Image
%
% Output:
%   -MHI: Updated MHI
%
% Dates:
%   -First Published: 1-Sept-2009

function [MHI] = UpdateMotionHistory(MHI, silhouette, timestamp,
DELTA)

    %Update all silhoutte location with current timestamp
    MHI(silhouette) = timestamp;

    %Set all MHI location to 0, if timestamps in given locations are
older than DELTA
    keepMHIUpto = timestamp-DELTA; %Either we could do (timestamp-
DELTA-1)
    %OR we could always assign DELTA value to 1 less
    toZero = MHI < keepMHIUpto;
    MHI(toZero) = 0;

    clear keepMHIUpto toZero

end %END: function [MHI] = UpdateMotionHistory(MHI, silhouette,
timestamp, DELTA)
```

A.24 GetMotionGradientOrientations.m

```
% function [orientation] = GetMotionGradientOrientations(MHI,
% DELTA_MIN, DELTA_MAX,GRADIENT_EPSILON,ONE_EIGHTY_BY_PI)
%
%     Author: Rudra PK Poudel
%     Contact: rudrapoudel@gmail.com
%     (c) Copyright University of Sussex
%
% Generate the Motion Gradient Orientations from the give MHI.
%
% Input:
%   -MHI: MHI to generate the Motion Gradient Orientation
%
%   -DELTA_MIN: Set the MGO to 0 if value of neighbour is less than
%   Delta_Min
%
%   -DELTA_MAX: Set the MGO to 0 if value of neighbour is greater than
%   Delta_Max
%
%   -GRADIENT_EPSILON: Set the MGO to 0 if value of X or Y gradient is
less
%   than Gradient_Epsilon
%
%   -ONE_EIGHTY_BY_PI: Value of 180.PI to optimize the code
%
% Output:
%   -orientation: Motion Gradient Orientation
%
% Dates:
%   -First Published: 1-Sept-2009

function [orientation] = GetMotionGradientOrientations(MHI, DELTA_MIN,
DELTA_MAX,GRADIENT_EPSILON,ONE_EIGHTY_BY_PI)

    %Calculating the X and Y derivatives using Sobel operator
    h = fspecial('sobel');
    Fy = conv2(MHI,h,'same'); %Y-Derivative

    h = -h';
    Fx = conv2(MHI,h,'same'); %X-Derivative

    % Calculating orientation using X and Y derivatives
    %orientation = rad2deg(atan(Fy ./ Fx)); %Gradient orientation
    orientation = ( atan(Fy ./ Fx) * ONE_EIGHTY_BY_PI ); %Gradient
orientation

%     %Allocating the mask, by default all are on
%     mask = MHI .* 1;

    %off the mask where X and Y both gradients are too low
    toZero = (abs(Fx) < GRADIENT_EPSILON) & (abs(Fy) <
GRADIENT_EPSILON);
%     mask(toZero) = 0;
    orientation(toZero) = 0;

    %Also off the mask on the border and noise area, using dilate and
erode
```

```
    %trick
    se = strel('square',3);
    Fy = imdilate(MHI,se,'notpacked','same');
    Fx = imerode(MHI,se,'notpacked','same');

    tempFyMinusFx = Fy - Fx;
    toZero = tempFyMinusFx < DELTA_MIN | tempFyMinusFx > DELTA_MAX;
%     mask(toZero) = 0;
    orientation(toZero) = 0;

    clear h Fy Fx toZero se tempFyMinusFx

end %END: function [orientation] = GetMotionGradientOrientations(MHI,
DELTA_MIN, DELTA_MAX,GRADIENT_EPSILON,ONE_EIGHTY_BY_PI)
```