CHAPTER 4

Generalized Active Energy Image

In the previous chapter, i.e., Chapter 3, Pose-based BEI features have been constructed that encode information about the contours of the silhouette images at the granularity of key poses. However, a shortcoming of this method, as well as other existing pose-based gait recognition, approaches [1, 34, 58] is that their effectiveness depends on the accuracy of a pre-defined set of key poses and are, in general, not robust against varying walking speeds. In this chapter, we propose an improvement to the existing pose-based approaches by considering a dictionary of key pose sets with multiple key poses each of which stores information about a gait cycle using a varying number of frames. This dictionary is also generic and is constructed from a large set of subjects that may/may not include the subjects present in the gait recognition data set. Comparison between a pair of training and test sequences is done by mapping the frames in each of these sequences into the individual key pose sets present in the above gallery set followed by computing contour-based features for each key pose, and next observing the frequency of matched key poses in all the sets. The main contributions of this work can be summarized as follows:

- The proposed gait analysis scheme improves over the existing pose-based gait recognition approaches in the sense that it can handle frame rate variation, and/or walking speed variation in a better manner.

- We construct a dictionary of key pose sets and a pair of training and test sequences is compared by finding a matching score between each key pose present in the different key pose sets and aggregating these scores.

As in the previous chapter, here also our approach has been evaluated using the scenarios given in Table 2.4 corresponding to the two popular gait data sets, namely the CASIA B data and the TUM-GAID data.

4.1 Overview

The block diagram of Fig. 4.1 shows the steps of our proposed approach. With reference to the block diagram, first preprocessing steps are performed and a complete gait cycle is extracted from each input sequence, as discussed in Chapters 1 and 3. Next, an optimal number of key poses is determined from each of these sequences. The difference of our work from the previous pose-based gait recognition approaches is that here the gallery sequences are not constrained to map to the same pre-computed key pose set. Rather, multiple key pose sets with a different number of key poses are computed from each sequence. Next, sequences with the same number of key poses are grouped together, and for each of these groups a set of generic key poses is computed by averaging the silhouettes that are mapped to the individual key walking stances. The different key pose sets obtained from all the above-mentioned groups are collectively termed as *Dictionary of Key Pose Sets*. Determination of the class of an input test subject is done by computing the *Generalized Active Energy Image* (GAEI) features from the dictionary. The individual blocks shown in the block diagram are explained in further detail in the subsequent sub-sections.

4.1.1 Extraction of Key Walking Stances from Each Sequence and Forming the Dictionary of Key Pose Sets

As in any key pose-based gait recognition method, an input sequence is mapped to the appropriate key poses by satisfying the following constraints:

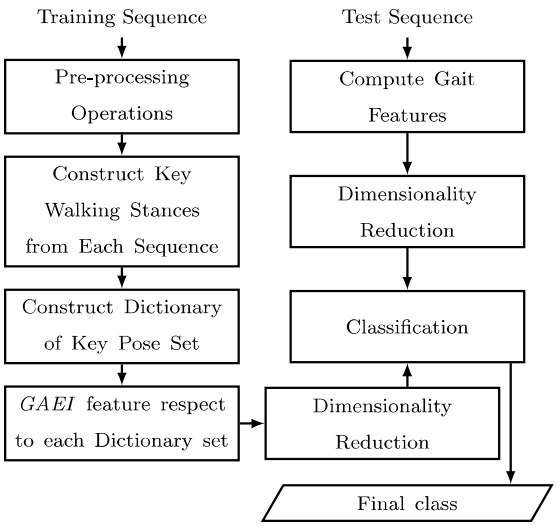


Figure 4.1: Block diagram of the proposed gait recognition approach

- At least one frame in a half gait cycle must map to each pose present in the set of key walking poses.
- The temporal ordering of the frames in a gait sequence of a subject must be exactly similar to that of the key pose sequence.

However, the variability in the walking speed of individual persons is not well captured by these approaches due to the strict assumption of the first constraint. Some people have a natural tendency to walk at a brisk pace, while some others may walk at a slow pace, and the key walking stances for these two types of walking should be differently modelled. As an improvement, we propose determining several key pose sets of varying cardinality (i.e., number of key poses is each set is different) to represent the different speeds of walking. This set of key poses has been termed as the *Dictionary of Key Pose Sets*. To come up with a generic representation of this dictionary, the gallery set must be chosen to be sufficiently large and should ideally be different from the gallery set for gait recognition. In this work, we consider several half gait cycles from the CASIA B [2] and the TUM-GAID [3] data to construct the dictionary of key pose sets. Estimating a key pose set from each sequence is done by applying constrained K-Means clustering on

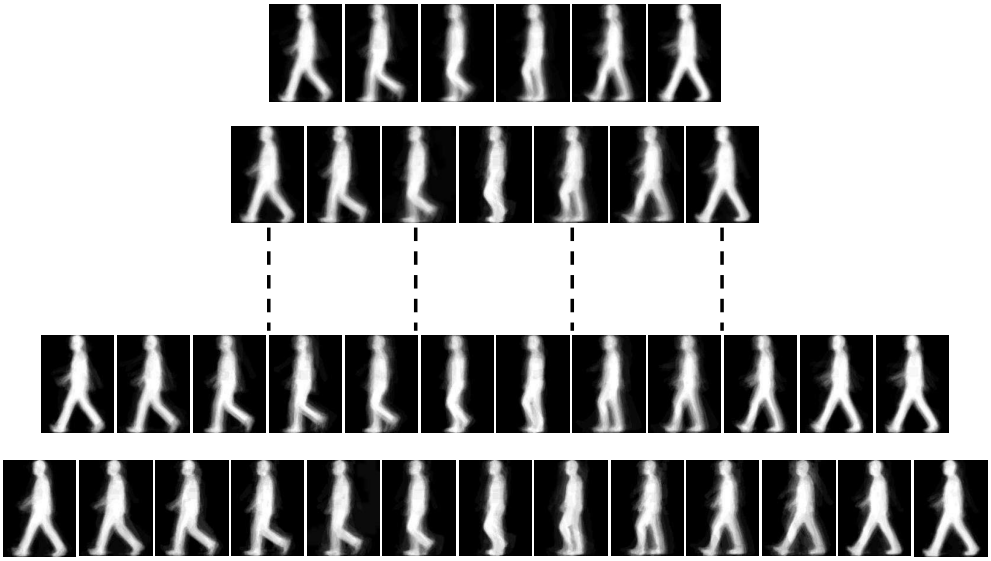


Figure 4.2: Key pose sets in the dictionary with cardinality from 6 to 13 where each set starts from double-support and ends with same pose

the frames present in the sequence multiple times for different values of K. The two constraints imposed on the constrained clustering step are as follows:

- If a frame is mapped to the k^{th} cluster, the next frame must be mapped to either the k^{th} or the $(k+1)^{th}$ cluster.
- After completing the clustering process taking into consideration the above constraint, the transition order of each frame is checked. If the frames are not ordered properly, re-assignment of cluster indices is done so that the previous frame's cluster index is less than or equal to the present frame's cluster index.

Figure 4.2 shows an example of the dictionary of key pose sets with cardinality from 6 to 13 obtained by considering the K values from 6 to 13, respectively. As can be seen from the figure, each row of the dictionary corresponds to half of a gait cycle starting and ending with the double support stance pose. Let S_1, S_2, ..., S_M denote a set of M sequences present in the gallery set for forming the dictionary of key pose sets. Further, assume that P^{S_i} denote the key pose set corresponding

to the sequence S_i, and d^{S_i} be the cardinality of the sequence, i= 1, 2, ..., M. Mathematically:

$$P^{S_1} = \{p_1^{S_1}, p_2^{S_1}, ...p_{d^{S_1}}^{S_1}\}$$
$$P^{S_2} = \{p_1^{S_2}, p_2^{S_2}, ...p_{d^{S_2}}^{S_2}\}$$
$$P^{S_M} = \{p_1^{S_M}, p_2^{S_M}, ...p_{d^{S_M}}^{S_M}\}$$

The optimal number of key poses obtained from all the sequences present in the gallery will not necessarily be the same. However, it may be assumed that two sequences from which same number of key poses are obtained bear significant similarity in their walking characteristics. This is since any gait cycle must consist of a certain set of representative poses given by the key poses and for a given frame rate if two subjects cover all these poses within a fixed amount of time, then it is highly likely that the two subjects bear significant similarity in their walking speed and pose transitions.

Generic representations of the different key pose sets are next obtained by averaging the key pose sets with same cardinality, and these together form the Dictionary of Key Pose Sets. Let us assume that this dictionary consists of a total of N averaged key pose sets, where $N < M$. Among these, let us consider the j^{th} element (say, E_j) that is formed by averaging t key pose sets with same cardinality (say, K_j). Further, suppose that the key pose sets constituting the dictionary element E_j are represented as $P^{S_{j_1}}$, $P^{S_{j_2}}$, ..., $P^{S_{j_t}}$, respectively (j=1,2,...,N). Mathematically,

$$E_j = \frac{1}{t}\sum_{y=1}^{t} P^{S_{j_y}}, = \{\frac{1}{t}\sum_{y=1}^{t} p_1^{S_{j_y}}, \ \frac{1}{t}\sum_{y=1}^{t} p_2^{S_{j_y}}, \ ..., \ \frac{1}{t}\sum_{y=1}^{t} p_{K_j}^{S_{j_y}}\}. \tag{4.1}$$

If this dictionary is denoted by E, then

$$E = \{E_1 \ E_2 \ ... \ E_N\}, \tag{4.2}$$

Construction of the Dictionary of Key Pose Sets, as discussed above, is also presented in Algorithm 1.

Algorithm 1: Construction of the Dictionary of Key Pose Sets

Input: M different gait cycles S_1, S_2, ..., S_M, each starting with a double support stance.

Start

Step 1: Set $i \longleftarrow 1$.

Step 2: **while** $i < M$ **do**

  Step 2.1: Consider the i^{th} gait cycle S_i, and divide it into optimal number of partitions d^{S_i} by following an approach similar to that described in [1, 58].

  Step 2.2: Form key pose set $P^{S_i} = \left\{p_1^{S_i}, p_2^{S_i}, ...p_{d^{S_i}}^{S_i}\right\}$ by averaging silhouettes that are mapped to each key pose.

  Step 2.3: $i = i+1$.

end

Step 3: The above key pose sets computed from M sequences are now grouped based on cardinality. Suppose, there exists N such groups with cardinalities K_1, K_2, ... K_N.

Step 4: Set $j \longleftarrow 1$.

Step 5: **while** $j < N$ **do**

  Step 5.1: Let $P^{S_{j_1}}$, $P^{S_{j_2}}$, ..., $P^{S_{j_t}}$ among the pre-determined key pose sets have cardinality K_j.

  Step 5.2: Form Dictionary Element E_j of cardinality K_j by averaging individual elements of the above t key pose sets (refer to Eqn. 4.1).

  Step 5.3: $j = j+1$.

end

Step 6: Aggregate all the dictionary elements to form the Dictionary of Key Pose Sets.

End

4.1.2 Extraction of the Generalized Active Energy Image Descriptor

Let us assume that the gallery set for gait recognition consists of $\mathcal{M}$ sequences denoted by $\mathcal{S}_1$, $\mathcal{S}_2$, ..., $\mathcal{S}_\mathcal{M}$, respectively. Let the classes (i.e., the subject identities) corresponding to these $\mathcal{M}$ sequences be respectively denoted by C_1, C_2, ..., $C_\mathcal{M}$. Next, gait features for each subject are generated corresponding to each of the N key pose sets present in the pre-constructed dictionary. Suppose, that a particular gallery sequence $\mathcal{S}_i$ consists of T frames, say F_1, F_2, ..., F_T. Also consider mapping of each frame of this sequence to the j^{th} key pose set E_j. Since Euclidean distance-based mapping of frames to key poses fails to preserve the temporal aspects of walking, we make use of the temporal constraints imposed by a state-transition diagram for this purpose as also discussed in Section 3.2.1 of Chapter 3.

Suppose, the frames F_1, F_2, ..., F_T of the input sequence are mapped to key poses corresponding to the dictionary element E_j. If this has K_j number of key poses, and the key pose indices to which the frames of the input sequence get mapped be respectively denoted by p_1, p_2, ..., p_T, then $p_k \leq K_j$, $\forall$ k = 1,2,..., T. Thus, the key pose indices p_1, p_2, ..., p_T are not all unique and multiple frames can be mapped to the same key pose. Figure 4.3(a) shows some frames from an input binary sequence and the corresponding key pose index to which each frame of the sequence is mapped. Consider, a key pose set of cardinality six and the mapping order of the frames of an input sequence to these six key poses is shown in Figure 4.3(a). Key pose-based Active Energy Image (AEI) [20] is next computed for each of the distinct key poses. For this, we construct a difference image sequence by computing pixel-wise absolute differences between adjacent silhouettes. If the frames in the difference image sequence are denoted by U_1, U_2, ..., U_{T-1}, then,

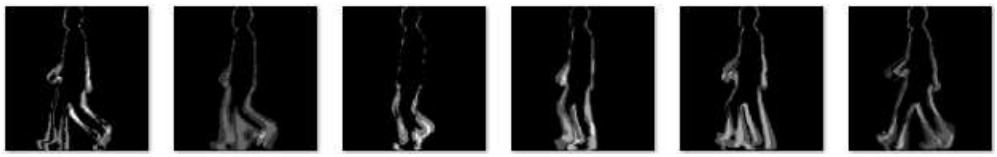
$$U_t = |F_{t+1} - F_t|, \forall t = 1, 2, ..., (T-1). \tag{4.3}$$

Fig. 4.3(b) shows the difference images obtained by applying (4.3) on the above sequence. The last frame in Fig. 4.3(b) corresponds to the absolute difference between the last frame shown in Fig. 4.3(a) and the subsequent frame of the same sequence (not shown here).

The frames in the difference image sequence corresponding to which the frames in

(a)

(b)



(c)

Figure 4.3: (a) An input sequence along with the key pose indices obtained after mapping the sequence to a dictionary element of cardinality six, (b) Difference image sequence obtained from the above binary sequence, (c) Generalized Active Energy Image obtained from the above mapping

the binary silhouette sequence are mapped to the same key pose are next averaged to generate the AEI for that key pose. Let $\mathcal{A}^j_{i,k}$ denote the AEI feature computed from the difference images corresponding to the frames F_{t_1}, F_{t_2}, ..., F_{t_n} of the input sequence $\mathcal{S}_i$ that got mapped to key pose p_k. Here, $1 \leq \{t_1, t_2, ..., t_n\} \leq T$, and $1 \leq j \leq N$. Corresponding to the K_j key poses in the dictionary element E_j, we extract K_j AEIs in a similar manner, and we repeat this process for all the N elements (i.e., the averaged key pose sets) present in the dictionary of key poses. The above process is explained step-wise using Algorithm 2.

Algorithm 2: Pose-based Active Energy Feature Construction from a Specific Key Pose Set in the Pre-Constructed Dictionary

Input: The j^{th} element of the Dictionary of Key Pose Sets E_j and an input gait sequence $\{F_1, F_2, \ldots, F_T\}$ of i^{th} subject.
Start
Step 1: Consider the dictionary element E_j and map each frame of the input gait sequence into the appropriate key poses present in E_j using the state transition model (refer to Fig. 3.7 of Chapter 3).
Step 2: Construct the difference frame sequence U_1, U_2, ..., U_{T-1} by computing absolute differences between consecutive frames of the input sequence (refer to Eqn. 4.3).
Step 3: Compute pose-based Active Energy Image from the difference image sequence. Since, dictionary element E_j consists of K_j key poses, we derive K_j Active Energy Images, denoted by $\mathcal{A}^j_{i,1}$, $\mathcal{A}^j_{i,2}$, ..., $\mathcal{A}^j_{i,K_j}$.
End

The Generalized Active Energy Image ($GAEI$) descriptor for a sequence $\mathcal{S}_i$ is henceforth represented as a set of tuples of feature sets, where each tuple corresponds to the AEI features extracted from a particular key pose set in the dictionary. If the descriptor is denoted by G_i, then,

$$G_i = \{\{\mathcal{A}^1_{i,1}, \mathcal{A}^1_{i,2}, ..., \mathcal{A}^1_{i,K_1}\}, \{\mathcal{A}^2_{i,1}, \mathcal{A}^2_{i,2}, ..., \mathcal{A}^2_{i,K_2}\}, ..., \{\mathcal{A}^N_{i,1}, \mathcal{A}^N_{i,2}, ..., \mathcal{A}^N_{i,K_N}\}\}. \quad (4.4)$$

The $GAEI$ features corresponding to all the training sequences $\mathcal{S}_1$, $\mathcal{S}_2$, ..., $\mathcal{S}_\mathcal{M}$ form a gallery database of $\mathcal{M}$ features. If this set is denoted by $\mathcal{G}$, then the ordered

pairs of features and target classes in this set are given by:

$$\mathcal{G} = \{(G_1, C_1), (G_2, C_2), ..., (G_{\mathcal{M}}, C_{\mathcal{M}})\}, \tag{4.5}$$

where G_i and C_i respectively denote the *GAEI* features and class of the sequence $\mathcal{S}_i$, as described before. Fig. 4.3(c) shows the *GAEI* images obtained from the silhouette sequence shown in Fig. 4.3(a).

4.1.3 Classification of a Probe Subject

Consider a particular probe sequence $\mathcal{S}_z$, and let the *GAEI* features computed from this sequence be denoted by $\hat{G}$, where

$$\hat{G} = \{\{\hat{\mathcal{A}}_1^1, \hat{\mathcal{A}}_2^1, ..., \hat{\mathcal{A}}_{K_1}^1\}, \{\hat{\mathcal{A}}_1^2, \hat{\mathcal{A}}_2^2, ..., \hat{\mathcal{A}}_{K_2}^2\}, ..., \{\hat{\mathcal{A}}_1^N, \hat{\mathcal{A}}_2^N, ..., \hat{\mathcal{A}}_{K_N}^N\}\}. \tag{4.6}$$

To find the class of the test sequence, we compare $\hat{G}$ with each of G_1, G_2, ..., G_N, and find the closest match. For this, we predict the probable class of the test sequence from the individual attributes of the *GAEI* feature. Basically, the class of $\hat{\mathcal{A}}_1^1$ is predicted from gallery features $\mathcal{A}_{1,1}^1$, $\mathcal{A}_{2,1}^1$, $\mathcal{A}_{\mathcal{M},1}^1$, class of $\hat{\mathcal{A}}_2^1$ is predicted from $\mathcal{A}_{1,2}^1$, $\mathcal{A}_{2,2}^1$, $\mathcal{A}_{\mathcal{M},2}^1$, and so on. If L denotes the dimensionality of the *GAEI* feature, then

$$L = K_1 + K_2 + ... + K_N. \tag{4.7}$$

Instead of using the original gallery gait features for the above comparison, we first pass the features through a fully-connected Autoencoder neural network to get a representation in a reduced space. This network can perform a non-linear transformation of the input features into a reduced space unlike PCA which essentially performs linear transformation only.

4.1.3.1 Description of Autoencoder Neural Network

An Autoencoder is a neural network that can be trained to replicate its input at its output, and thus, the training of an Autoencoder is unsupervised. The network is composed of two sub-networks: an encoder and a decoder, where both the sub-networks can have multiple layers stacked one after another in a fully-connected

manner. In this work, we consider both encoder and decoder networks to have one layer. If the input to an Autoencoder is a vector $x \in R^{D_x}$, then the encoder maps the vector x to another vector $z \in R^{D_z}$ as follows:

$$z = h^{(1)}(W^{(1)}x + b^{(1)}). \tag{4.8}$$

Here, the superscript (1) indicates the first layer. $h^{(1)}$: $R^{D_x} \rightarrow R^{D_z}$ is a transfer function for the encoder, $W^{(1)} \in R^{D_z \times D_x}$ is a weight matrix, and $b^{(1)} \in R^{D_z}$ is a bias vector. Then, the decoder maps the encoded representation z back into an estimate $\hat{x}$ of the original input vector x as follows:

$$\hat{x} = h^{(2)}(W^{(2)}z + b^{(2)}). \tag{4.9}$$

Here, the superscript (2) represents the second layer. $h^{(2)}$: $R^{D_z} \rightarrow R^{D_x}$ is a transfer function for the encoder, $W^{(1)} \in R^{D_x \times D_z}$ is a weight matrix, and $b^{(2)} \in R^{D_x}$ is a bias vector. The above description provides a general information about the unsupervised Autoencoder. The Autoencoder used in this work is a Sparse Autoencoder with Sparsity Regularizer and L2 Regularizer [112]. A sparse Autoencoder allows the network to sensitize individual hidden layer nodes toward specific attributes of the input data, whereas an under-complete Autoencoder will use the entire network for every observation. Thus for sparse Autoencoder, we have limited the capacity of the network to memorize the input data without affecting the capability of the network to extract features from the data. This allows us to consider the latent state representation and regularization of the network separately, such that we can choose a latent state representation (i.e., encoding dimension) according to the context of the data while imposing regularization by the sparsity constraint.

4.1.3.2 Sparse Autoencoder

Encouraging the sparsity of an autoencoder is possible by adding a regularizer to the cost function [112]. This regularizer is a function of the average output activation value of a neuron. The average output activation measure of a neuron

i defined by $\hat{\rho}_i$ is computed as:

$$\hat{\rho}_i = \frac{1}{n}\sum_{j=1}^{n} z^{(1)}(x_j) \tag{4.10}$$

where $z^{(1)} = h^{(1)}(W^{(1)T}x_j + b^{(1)})$, n being the total number of training examples, x_j the j^{th} training example, $W^{(1)}$ the weight matrix, and $b^{(1)}$ the bias vector. A neuron is considered to be 'firing' if its output activation value is high. A low output activation value means that the neuron in the hidden layer fires in response to a small number of the training examples. Adding a term to the cost function that constrains the values of $\hat{\rho}_i$ to be low encourages the Autoencoder to learn a representation, where each neuron in the hidden layer fires to a small number of training examples. In other words, each neuron specializes by responding to some feature that is only present in a small subset of the training examples.

4.1.3.3 Sparsity Regularization

Sparsity regularizer attempts to enforce a constraint on the sparsity of the output from the hidden layer. Sparsity can be encouraged by adding a regularization term that takes a large value when the average activation value, $\hat{\rho}_i$, of neuron i and its desired value, ρ, are not close in value [112]. One such sparsity regularization term can be the Kullback-Leibler divergence.

$$\Omega_{sparsity} = \sum_{i=1}^{D^{(1)}} KL(\rho||\hat{\rho}_i) = \sum_{i=1}^{D^{(1)}} \rho \log(\frac{\rho}{\hat{\rho}_i}) + (1-\rho)\log(\frac{1-\rho}{1-\hat{\rho}_i}) \tag{4.11}$$

Kullback-Leibler divergence is a function for measuring how different two distributions are. In this case, it takes the value zero when ρ and $\hat{\rho}_i$ are equal to each other and becomes larger as they diverge from each other. Minimizing the cost function forces this term to be small, hence ρ and $\hat{\rho}_i$ to be close to each other. You can define the desired value of the average activation value using the Sparsity Proportion name-value pair argument while training an Autoencoder.

4.1.3.4 L2 Regularization

When training a sparse Autoencoder, it is possible to make the sparsity regularizer small by increasing the values of the weight components $w^{(l)}$ and decreasing the values of $z^{(l)}$ [112]. Adding a regularization term on the weights to the cost function prevents it from happening. This term is called the L2 regularization term and is defined by:

$$\Omega_{weights} = \frac{1}{2}\sum_{l=1}^{H}\sum_{j=1}^{n_l}\sum_{i=1}^{k_l}(w_{ji}^{(l)}) \tag{4.12}$$

where H is the number of hidden layers, n_l is the output size of layer l, and k_l is the input size of layer l. The L_2 regularization term is the sum of the squared elements of the weight matrices for each layer.

4.1.3.5 Cost Function

The cost function for training a sparse Autoencoder is an adjusted mean squared error function as follows:

$$E = \frac{1}{N}\sum_{n=l}^{N}\sum_{k=l}^{K}(x_{kn} - \hat{x}_{kn})^2 + \lambda * \Omega_{weights} + \beta * \Omega_{sparsity} \tag{4.13}$$

where λ is the coefficient for the L_2 regularization term and β is the coefficient for the sparsity regularization term. The values of λ and β are specified using the L2-WeightRegularization and SparsityRegularization name-value pair arguments respectively while training an Autoencoder. The hyper-parameter for training the Autoencoder are as follows: hiddenSize = 362, MaxEpochs = 5, L2WeightRegularization = 0.004, SparsityRegularization = 4, SparsityProportion = 0.15, which has been seen to perform with highest accuracy on a validation set among some other hyper-parameter configurations chosen for our experiments. Next, the test features are also projected into this same space and a *Linear Discriminant Analysis* classifier is finally used to classify each test attribute into the appropriate class.

For sequences corresponding to the same identity in the training and test sets, the *GAEI* features corresponding to the different key poses are expected to be similar

compared to that for different identities. Even if there are a few mismatches due to the presence of noise in the training or test frames, incorrect background subtraction, or slight variations in the walking conditions during capturing the training and test videos at two different times, a combined prediction from the different key poses through majority voting is expected to provide the correct class. Hence, to predict the class of the test subject from the L classification results, we adopt the following procedure. First, we find out the unique predicted classes and next sum up the classification probabilities for these classes. Thus, if the L classification results consist of K unique classes, C_{k_1}, C_{k_2}, ..., $C_{k_{\mathbb{K}}}$, and if the sum of the probabilities for these classes are respectively denoted by $\mathcal{P}_{k_1}$, $\mathcal{P}_{k_2}$, ..., $\mathcal{P}_{k_{\mathbb{K}}}$, then the test subject is assigned class C_{j*} if

$$\mathcal{P}_{j*} < \mathcal{P}_j, \forall j \in \{k_1, k_2, ..., k_{\mathbb{K}}\}, \;\; j* \neq j. \tag{4.14}$$

4.2 Experiments and Results

The same system used to perform the experiments in Chapter 3 has also been employed here. To train the Autoencoder model, out of the four normal walking sequences in CASIA B data, namely *nm-03*, *nm-04*, *nm-05*, and *nm-06*, we choose two at a time as training set and remaining two as validation set. Like-wise, we obtain six different pairs of training and validation sets, and the average cross-validation accuracy obtained from all the validation sets is plotted in Fig. 4.4 by varying the number of hidden layer neurons from 100 to 1000 in steps of 100. A similar experiment is repeated again for the TUM-GAID data, and the average cross-validation accuracy values for this data are plotted in the same figure. From the figure, we decide the number of hidden neurons to be 400, since after this point the curve attains an almost saturation level in both the cases, and increasing the number of hidden neurons beyond 400 is likely to overfit the training data.
Next, we make a comparative analysis in terms of rank 1 accuracy between our proposed approach with other existing popular gait recognition methods working with complete gait cycle information, namely those using GEI [31], PEI [1], AEI [51], EAI [113], an approach using Deep CNN [49] with two different batch sizes (16 and 24), GEINet-based recognition [91], a signal processing-based approach

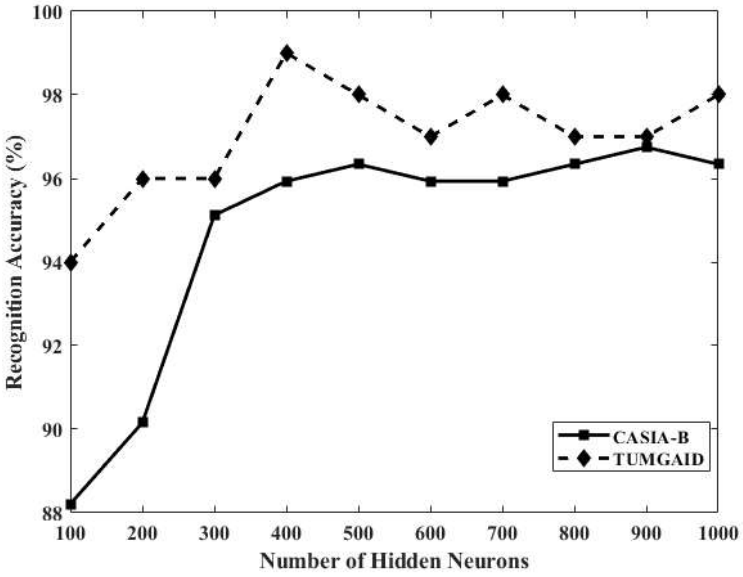


Figure 4.4: Effect of Increasing Hidden Layer Neurons of the auto-encoder on the classification accuracy of CASIA B and TUM-GAID data

[69], a pose-based approach that uses Deep Learning [114], and also the approach using Pose-based *BEI*, as discussed in Chapter 2. Cross-view gait recognition methods [95, 96] and those dealing with partial gait cycle information [115, 116] have not been used in this study since these are not related to the main theme of our work, i.e., developing a generalized pose-based recognition scheme that can be conveniently integrated with other gait features derived from video sequences captured from the same view as that of the training sequences.

Although, we propose integrating the popular AEI feature [51] with the new pose-based recognition scheme (due to its capabilities in preserving the dynamic characteristics of gait better than GEI or other feature aggregation-based methods), in the present experiment, we also study the results obtained by integrating the GEI feature [31] with the proposed scheme. Results are shown in Tables 4.1 and 4.2 for the CASIA B and the TUM-GAID data, respectively. Each of the different training-test scenarios as discussed in Table 2.1 correspond to the second, third, fourth, fifth and sixth columns in both the tables. The first column represents the gait feature, while the last column depicts the mean accuracy of each gait feature

for the different training-test criteria. The trained Autoencoder models for the two data sets, as obtained from the previous experiments, have also been used to get the results of the present experiment.

Table 4.1: Comparative performance analysis of the different gait recognition methods on CASIA B data [2] in terms of Rank 1 accuracy

Approach	C_1 (%)	C_2 (%)	C_3 (%)	C_4 (%)	C_5 (%)	**Mean** (%)
GEI [27]	**99.59**	95.12	98.37	68.04	34.55	79.20
AEI [20]	99.59	82.25	94.35	88.61	45.93	78.04
EAI [113]	**99.59**	91.05	93.49	83.08	45.52	82.54
DeepCNN [49] batch size 16	97.15	81.30	84.55	35.77	19.51	63.65
DeepCNN [49] batch size 24	97.96	82.92	89.43	28.04	15.85	62.84
GEINet [91]	96.74	76.42	83.73	23.17	10.97	58.21
DTW-Signal [69]	94.71	86.99	86.99	56.50	27.64	70.56
Deep-Pose [114]	92.68	85.36	72.35	72.60	41.82	72.96
MGANs [50]	99.59	84.89	82.04	91.00	48.00	81.10
GaitSet [94] LT	91.70	80.65	82.88	81.00	**70.10**	81.26
GaitGAN [95]	98.39	81.70	80.86	64.52	48.39	74.77
PEI [1] with 6 key poses	**99.59**	88.70	97.58	78.04	43.90	81.56
BEI	98.38	87.90	89.51	90.32	29.43	79.10
Pose based BEI with 6 poses	**99.59**	81.45	90.32	**92.33**	57.25	84.56
Our approach using GEI	97.96	**100.00**	**99.18**	81.70	44.71	84.71
***GAEI* (proposed)**	98.78	99.18	**99.18**	90.24	**66.66**	**90.80**

In Tables 4.1 and 4.2, the best accuracy values obtained for each of the scenarios C_1, C_2, ..., C_5 in case of the CASIA B data, and T_1, T_2, ..., T_5 in case of the TUM-GAID data (as shown in Table 2.1) have been shown in bold fonts. It is seen that generally each of the existing techniques performs quite well if the test sequence is devoid of co-variate factors such as carrying bag, wearing coat, etc. (refer to the 5^{th} and 6^{th} columns of both the tables). If the training and test conditions are made different by introducing co-variate conditions, the accuracy of these methods is not reliable.

Table 4.2: Comparative performance analysis of the different gait recognition methods on TUM-GAID data [3] in terms of Rank 1 accuracy

Approach	T_1 (%)	T_2 (%)	T_3 (%)	T_4 (%)	T_5 (%)	**Mean** (%)
GEI [27]	96.54	85.52	86.18	25.16	85.36	75.75
AEI [20]	91.14	38.36	22.95	**72.78**	79.01	60.82
EAI [113]	96.21	42.76	45.06	69.90	86.18	68.02
DeepCNN[49] batch size 16	88.48	39.14	24.67	21.54	66.11	47.98
DeepCNN[49] batch size 24	90.78	32.56	32.56	29.60	60.36	49.17
GEINet [91]	92.59	21.05	32.56	39.14	76.15	52.30
DTW-Signal [69]	65.90	25.24	28.19	12.29	59.67	38.25
Deep-Pose [114]	87.89	38.94	38.61	67.56	76.44	61.89
MGANs [50]	95.23	57.99	61.58	66.61	85.19	73.32
GaitSet [94] LT	86.18	50.02	56.07	58.55	76.15	65.39
GaitGAN [95]	94.57	41.50	58.82	64.27	83.22	68.47
PEI [1] with 6 key poses	93.09	92.76	89.80	17.76	84.04	75.49
BEI	93.93	67.54	62.29	68.29	81.47	74.75
Pose based BEI with 6 poses	90.32	55.72	44.58	62.62	77.86	66.22
Our approach using GEI	**96.71**	98.02	96.38	17.76	**88.65**	76.93
***GAEI* (proposed)**	76.48	**99.34**	**98.02**	51.15	65.13	**78.02**

For key pose-based gait recognition techniques, whenever co-variate conditions are added, the silhouette appearances undergo certain changes which causes incorrect mapping of frames to key poses, ultimately resulting in reduction of the prediction accuracy. On the other hand, gait recognition algorithms that take advantage of the dynamic walking information through adjacent frame differences, namely [51] and our proposed approach, show a more consistent performance compared to that of [1, 31, 49, 91, 95]. The ensemble-based method proposed in [113] also performs robustly for the different test conditions, but its average accuracy is significantly less compared to the proposed approach due to ignoring the dynamic aspects of motion. The few deep learning algorithms proposed in the recent years, e.g., [91, 114] have been seen to achieve a more reliable and consistent performance

against the varying co-variate conditions compared to the previous techniques. However, these methods compute features from the silhouette shapes only and ignore the dynamic information obtained from frame differences. In contrast, our proposed feature (namely, the Generalized Active Energy Image), takes advantage of both generalized pose-based feature representation, as well as dynamic information preservation, which has resulted in an improved recognition accuracy corresponding to both the data sets used in the study. Our approach further eliminates the dependency of the algorithm on the pre-determined set of key poses, which helps it to perform robustly against all the different co-variate conditions considered in Table 2.1. From the above experiment, it can be concluded that the proposed method outperforms each of the other competing approaches in terms of mean accuracy, and it also performs consistently well for each of the different settings of the training and testing conditions.

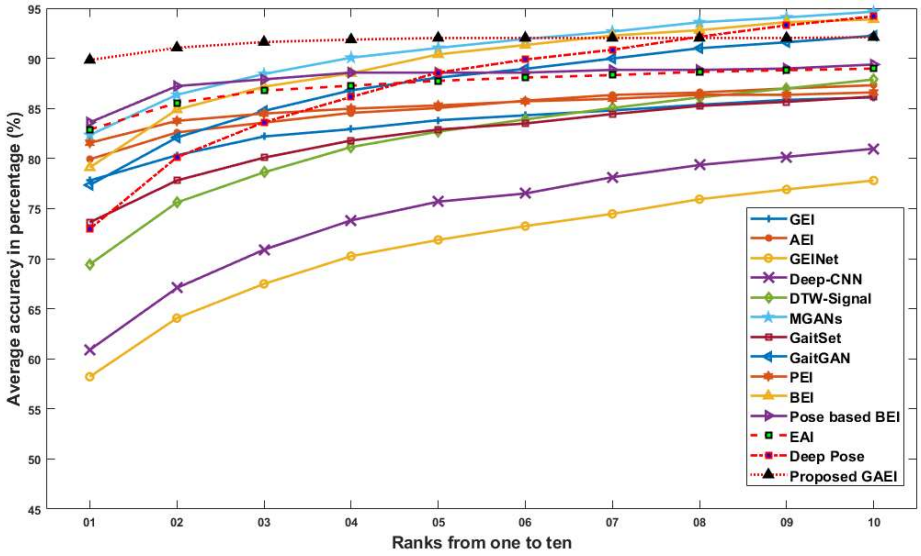


Figure 4.5: Comparison of rank-based accuracy with other gait recognition approaches using CASIA B

We next carry out a rank-based performance comparison of our method with that of the existing approaches in terms of average accuracy for both the CASIA B and TUM-GAID data sets. Similar to Chapter 3, for this experiment we consider the scenarios C_1, C_4, C_5 for the CASIA B data and T_1, T_4, T_5 for the TUM-

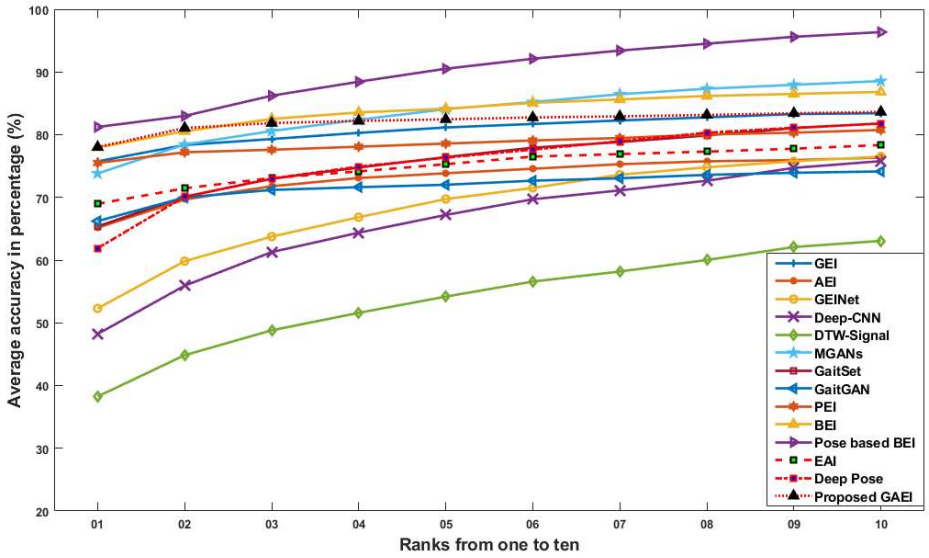


Figure 4.6: Comparison of rank-based accuracy with other gait recognition approaches using TUM-GAID data

GAID data. Since the deep learning-based approach in [49] shows very less rank 1 accuracy compared to the other approaches on an average, it has not been used in the present study. Corresponding results are shown using Cumulative Match Characteristic (CMC) curves in Figs. 4.5 and 4.6, respectively. In each of these figures, the accuracy values for the different methods are plotted along the vertical axis as the rank (plotted along the horizontal axis) is increased from 1 to 10. It can be seen from the Figs. 4.5 and 4.6 that our approach performs consistently well for both the datasets at the different rank values. For the CASIA B data, only a few approaches such as [50, 114] and the *BEI*-based method discussed in Chapter 3 provide slightly higher recognition accuracy values than that of our *GAEI*-based approach at high-rank values above 7. Till rank 6, our method performs the best among all the other approaches used in the present comparative study. On the other hand, for the Tum-GAID data, the approach using the Pose-based *BEI* feature proposed in Chapter 3 has been seen to perform better than our present approach for all the different rank values. However, each of the other compared methods performs either equivalently or with a lesser accuracy than our *GAEI*-based technique till rank 2. Similar to the observation from Fig. 4.5, once

again our previously proposed *BEI*-based method and that in [50] perform slightly better than our *GAEI*-based approach at higher rank values. The Pose-based *BEI* feature is made using a single set of unique poses where interior duplicate or redundant pixels are removed but *GAEI* performs better due to considering multiple sets of key poses of different lengths corresponding to a dictionary of key poses.

From the CMC curves in the Figs. 4.5 and 4.6, although the *BEI*-based method has been seen to provide a higher gait recognition accuracy than our present one, it may be noted that these CMC curves have been plotted considering the average performance corresponding to three scenarios only. From Tables 4.1 and 4.2, we can observe that our present method shows a more consistent and robust performance for both similar and varying co-variate conditions than most of the compared approaches including the *BEI*-based technique and that in [50, 114]. However, due to computing features for the several key pose sets in the dictionary and fusing this information to make the final prediction, the proposed *GAEI*-based gait recognition approach is time-intensive. We observe that for the CASIA B data which contains 124 subjects in the gallery, our method requires about 7.25 seconds to classify an individual as one among the gallery subjects.

4.3 Summary

The chapter introduces an appearance-based gait feature, termed *Generalized Active Energy Image (GAEI)*, for identifying a person from a gallery set of subjects using his/her gait video captured by an RGB camera. The *GAEI* feature captures the kinematic information of walking in the form of key poses of varying lengths. We improve over the existing pose-based gait recognition approaches and also the one proposed in the previous chapter of the thesis by reducing the dependency of pose-based gait recognition on the initially computed set of key poses. Our *GAEI*-based approach has been seen to perform accurately and consistently well for similar as well as varying co-variate conditions during training and testing. Experimental evaluation using two popular real-world data sets (CASIA B and TUM-GAID) verifies the effectiveness of our approach and also reveals its superiority over most other Deep Learning and non-Deep Learning-based gait recogni-

tion techniques. Our *GAEI*-based approach relies solely on hand-crafted features only and does not employ Deep Learning at any stage that makes it lightweight. However, it achieves a performance equivalent to a recent Deep Learning-based approach, namely, MGANs [50]. A limitation of the present method is that due to considering a dictionary of key poses, its response time is quite high, which makes it somewhat impractical for use in real-life scenarios where a fast response time is desirable. However, it can still be potentially employed to identify perpetrators in places where the gallery set and/or test set is expected to be less extensive such as in school or college environments. In surveillance sites such as airports, railway stations, and shopping malls the gallery and/or test set is usually very large and hence the computational overhead on applying this approach will also be quite high. It seems that the above limitation can be addressed well by keeping a single key pose set instead of a dictionary of key poses but relaxing the mapping constraints to a certain extent so that the shortcomings of the existing pose-based approaches can be handled in a more time-efficient manner.