Real-time Adaptive Hand Motion Recognition using a Sparse Bayesian Classifier

Shu-Fai Wong and Roberto Cipolla

Department of Engineering,
The University of Cambridge
{sfw26, cipolla}@cam.ac.uk

Abstract. An approach to increase adaptability of a recognition system, which can recognise 10 elementary gestures and be extended to sign language recognition, is proposed. In this work, recognition is done by firstly extracting a motion gradient orientation image from a raw video input and then classifying a feature vector generated from this image to one of the 10 gestures by a sparse Bayesian classifier. The classifier is designed in a way that it supports online incremental learning and it can be thus re-trained to increase its adaptability to an input captured under a new condition. Experiments show that the accuracy of the classifier can be boosted from less than 40% to over 80% by re-training it using 5 newly captured samples from each gesture class. Apart from having a better adaptability, the system can work reliably in real-time and give a probabilistic output that is useful in complex motion analysis.

1 Introduction

One of the challenges in building a system for sign recognition (and also gesture recognition) is that inter- and intra- personal *variation* may lead to a poor performance. In most situation, different signers may sign in different ways and even the same signer may not sign in the same way all the time (see Figure 4). A classifier that is capable to give a good classification result on one dataset may not be able to give a good result on another set. This idea can be illustrated by Figure 3 that shows an original decision boundary (the line in a lighter intensity) can separate original positive and negative samples ('$\times$' and '$\bullet$' in a lighter intensity) properly but not for new samples (shown in a darker intensity).

Adaptability is therefore an essential property of a sign recognition system applied on a wide range of users. Instead of training a recognition system using all possible samples, it is sensible to train the system using a limited amount of samples and then re-train it using online samples. As illustrated in Figure 3, an updated decision boundary (the line in a darker intensity) could be estimated based on some of the new samples in order to achieve a good classification result on both new and original samples.

In the past decade, sign recognition was done by exploiting *Hidden Markov Models* (HMMs). In [1], HMMs were directly applied to solve the problem and their extensions such as parallel HMMs [2] and self-organizing HMMs [3] were

also proposed to improve the performance. In addition, HMMs have also been extended to perform *adaptive* gesture recognition [4].

Recently, the use of HMMs has been criticised. One major criticism is that using HMMs to recognise gesture requires *large training sets* (e.g. [5]) and this will inhibit the *growth of the vocabulary*. In addition, recent HMMs design analyses *each sign as a whole* without breaking it down into corresponding components (e.g. [6]), making the model more complicated and reducing its *extensibility*. Although recent works (e.g. [5, 6]) provide some alternative solutions to the sign recognition problem and achieve an acceptable accuracy, they have not considered how to improve the adaptability of their recognition systems.

In this paper, an adaptive approach to recognise 10 *primitive movements*, which can be considered as the building blocks in any sign language system, is proposed. Motion recognition is done by exploiting *motion gradient orientation* (MGO) images to form motion features and using a *sparse Bayesian classifier* to map the features into their corresponding classes. The Bayesian classifier is designed in a way so that it can be *re-trained* using *online samples*. The present research has three main contributions. Firstly, by allowing online learning, the *adaptability* and the *accuracy* of the recognition system are raised. Secondly, due to the use of the Bayesian classifier, the final outcome is a *probabilistic* value, which is useful in high-level inference processes that must maintain multiple hypotheses. Thirdly, the classifier maintains a *sparse model*, which facilitates an efficient use of computational resources and leads to a real-time performance.

2 Approach

As mentioned in the previous section, HMMs are not the only choice for performing sign recognition and there are alternative solutions such as [5, 6]. This paper extends the work of Derpanis et al. [6] to allow online training and classifying inputs captured under a wider range of conditions. The basic framework and theory used will be described in the following sub-sections.

2.1 Framework

In [6], Derpanis et al. introduced the idea of *breaking down* signs into constituent *primitive movements* with the aid of linguistic information (e.g. [7]). Sign language recognition can then be considered as recognising the primitive movements and the corresponding sequence. Derpanis et al. used simple and manually defined mapping functions to map the motion data in time-series format into their corresponding movement classes. Their work is thus difficult to be extended. In this paper, we adopt their divide-and-conquer strategy but also exploit a general recognition procedure to increase the extensibility.

This paper focuses on the *hand motion classification* problem (i.e. classifying a given video sequence of hand motion into one of the movement primitives). Motivated by [6], we have 10 *primitive movements* to be classified: (1) upward, (2) downward, (3) rightward, (4) leftward, (5) toward signer, (6) away signer,

(7) nod, (8) supinate, (9) pronate, and (10) circular. Figure 1 illustrates these 10 primitives. The classification scheme used will be presented next in sub-section.

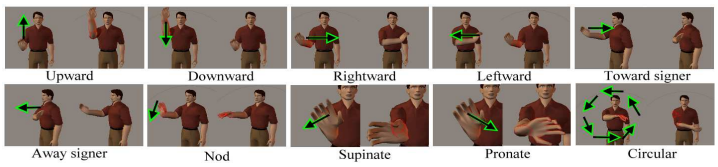

Fig. 1: *This figure shows the 10 primitive movements recognised by the proposed system.*

2.2 Theory

Unlike recent works such as [6, 5] that are exploiting simple classification schemes, this work is going to handle inter- and intra-personal variation through the use of a powerful classifier. Among several state-of-the-art classifiers, a *sparse Bayesian classifier* or *Relevance Vector Machine* (RVM) is used in the proposed system.

Compared with other state-of-the-art classifiers such as Support Vector Machine (SVM), the RVM classifier performs equally well in term of *accuracy*. In addition, the final outcome of the RVM classifier is a *probabilistic* value instead of a simple true-or-false answer. Furthermore, the *sparsity* of the model stored by the RVM classifier ensures fast and efficient classification process, and this implies the classifier can be implemented in computing devices with limited memory storage such as Pocket PC or Smartphone.

RVM classifier is a simple *binary classifier*. Consider a training set that consists of N motion feature vectors, $\{\mathbf{x_n}, t_n\}_{n=1}^{N}$. The problem of learning a binary classifier can be expressed as that of learning a function f so that the input feature $\mathbf{x_n}$ will map onto their correct classification label t_n and the probability of $\mathbf{x_n}$ is classified as the target class (where $t_n = 1$) equals to $\sigma(y_n) = 1/(1 + e^{-y_n})$where $y_n = f(\mathbf{x_n})$.

The function f can be written as a sparse model where ($M \ll N$) [8]:

$$f(\mathbf{x_n}) = \sum_{m=1}^{M} \omega_m \phi_m(\mathbf{x_n}) + \omega_0 \tag{1}$$

where $\omega = (\omega_0, ...\omega_M)^T$ are the weights and $\phi_m(\mathbf{x_n}) = K(\mathbf{x_n}, \mathbf{x_m})$ with $K(\cdot, \cdot)$ a positive definite kernel function (where Gaussian Kernel with width 1 is used in the proposed system) and $\mathbf{x_m}$ an example (or a relevance vector) from the training set. Under the RVM framework where hyperparameters $\alpha = \{\alpha_0, ..., \alpha_M\}$ are introduced, *learning* f from the training data means *inferring* ω from the data $\mathbf{t} = \{t_1, ..., t_N\}$ such that the posterior probability over the weights, $p(\omega \mid \mathbf{t}, \alpha)$, is maximised. Given $\mathbf{A} = \text{diag}(\alpha_0, \alpha_1, ..., \alpha_N)$, $\mathbf{B_{nn}} = \sigma\{y_n\}[1 - \sigma\{y_n\}]$ and $\mathbf{\Phi}$

is the $N \times (N+1)$ design matrix, the optimal values of the weights can be estimated by using an *iterative procedure* [8], where the inverse of a Hessian matrix at 'most probable' weight (ω_{MP}), $\nabla\nabla \log p(\mathbf{t}, \omega \mid \alpha)|_{w_{MP}} = -(\mathbf{\Phi^T B \Phi + A})$ have to be computed for a current, fixed values of α at each loop. The values of α can be inferred from the training data such that the marginal likelihood $p(\mathbf{t} \mid \alpha)$ is maximised. The iterative procedure for estimating ω and α is repeated until some suitable convergence criteria are satisfied.

In a *batch-learning* approach, all training examples will be considered as relevance vectors at the initial stage and the irrelevance vectors will be 'pruned' after re-evaluation of α in each iteration. In other words, every α_i has a finite value at the beginning and the Hessian matrix to be computed in each estimation loop has a size of $(N+1) \times (N+1)$ initially, where N is the number of training samples. Since inversion of Hessian matrices is involved in the learning algorithm, the overall *training complexity* is $O(N^3)$. This implies that if the initial sample size is huge, the learning algorithm may take a long time to converge.

According to [9], we can also start with an *initially small model* and sequentially 'add' relevance vectors to increase the marginal likelihood. Considering the marginal likelihood, or equivalently, its logarithm $\mathcal{L}(\alpha)$:

$$\mathcal{L}(\alpha) = \log p(\mathbf{t} \mid \alpha) = -\frac{1}{2}[N \log 2\pi + \log \mid \mathbf{C} \mid + \mathbf{t}^T \mathbf{C}^{-1} \mathbf{t}] \tag{2}$$

with $\mathbf{C} = \mathbf{B}^{-1} + \mathbf{\Phi A}^{-1} \mathbf{\Phi}^T$. From the analysis given in [9], $\mathbf{C}$ can be rewritten in this way: $\mathbf{C} = \mathbf{B}^{-1} + \sum_{m \neq i} \alpha_m^{-1} \phi_m \phi_m^T + \alpha_i^{-1} \phi_i \phi_i^T = \mathbf{C}_{-i} + \alpha_i^{-1} \phi_i \phi_i^T$, where $\mathbf{C_{-i}}$ is $\mathbf{C}$ without basis vector i. $\mathcal{L}(\alpha)$ can be therefore rewritten as:

$$\mathcal{L}(\alpha) = \mathcal{L}(\alpha_{-i}) + \frac{1}{2}[\log \alpha_i - \log(\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i}] \tag{3}$$

where $s_i = \phi_i^T \mathbf{C}_{-i}^{-1} \phi_i$ and $q_i = \phi_i^T \mathbf{C}_{-i}^{-1} \mathbf{t}$. From [9], estimation of α, which gives maximum value of marginal likelihood, can be computed directly from:

$$\begin{aligned} \alpha_i &= \tfrac{s_i^2}{q_i^2 - s_i}, && \text{if } q_i^2 > s_i, \\ \alpha_i &= \infty, && \text{if } q_i^2 \leq s_i, \end{aligned} \tag{4}$$

The implication of this evaluation method for α is that we can make discrete changes to the model while we are guaranteed to increase the marginal likelihood. This means we can start from an *initially small model* and *test* the 'relevance' of each new input vector i *sequentially*. When vector i is in the model (i.e. $\alpha_i < \infty$) but $q_i^2 \leq s_i$, then vector i should be removed (i.e. α_i set to ∞); When vector i is not in the model (i.e. $\alpha_i = \infty$) and $q_i^2 > s_i$, vector i should be added (i.e. α_i set to a certain optimal value). Classification model is thus built incrementally.

By adopting this *incremental training* approach, computational complexity is $O(M^3)$ where M is the number of relevance vectors and $M \ll N$. In other words, the *training time* can be *reduced* dramatically. In addition, this learning approach allows any new input to be *evaluated on the fly* and to be added to the model if certain criteria are fulfilled. This property can be used to develop an *online adaptive recognition system* where online training is needed.

3 Implementation details

The hand motion recognition problem addressed in this paper can be divided into 2 tasks, namely feature extraction and classification.

3.1 Feature Extraction

Extraction of MGO: In this work, a motion gradient orientation (MGO) image is extracted directly from a raw video input and transformed to a motion feature vector that contains necessary spatial-temporal information. MGO image was proposed by Bradski and Davis [10] to explicitly encode image changes introduced by motion events. The MGO is computed from a motion history image (MHI) and a motion energy image (MEI) [11]. MHI is an image that shows moving edges where the recency of a motion is represented by intensity level; MEI is a binary image that indicates where the current moving edges (moving regions) are; Pixels in MGO encode the change in orientation between nearest moving edges shown on the MHI and the region of interest is defined as the largest rectangle covering all bright pixels in MEI. The MGO therefore contains information about *where* and *how* a motion occurred. The MGO obtained within the region of interest will be rescaled to a standard size, which is 200×200 in the proposed system. Typical MGO images corresponding to the 10 primitive movements used are illustrated in Figure 2.

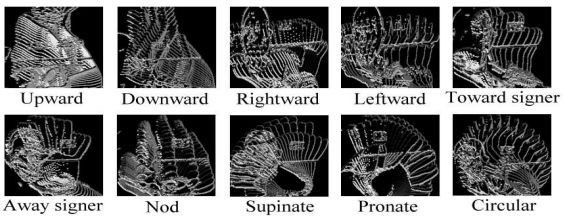

Fig. 2: *This figure illustrates the MGO images corresponding to the 10 primitive movements classified by the proposed system.*

Dimension Reduction: In order to reduce the necessary *number of training samples* (which is proportional to the dimension size), dimension reduction is done on the training MGO images that can be potentially very large in image size (i.e. the dimension size). Principal Component Analysis (PCA) is used in the proposed system to reduce the dimension of the MGO images. By performing PCA on all training MGO images, the eigenvalues indicate that the first 12 components provide an adequate summary of all the images, which account for 95% of the variation. Thus, the first 12 eigenvectors are chosen as the new basis functions for converting any new incoming MGO image into a new feature vector. Finally, normalization is done to give a final feature vector ($\mathbf{x}$) with zero mean and standard deviation of one in all dimensions.

3.2 Learning and Classification

Binary Classification: As explained in the previous section, given a training set $\{\mathbf{x_n}, t_n\}_{n=1}^{N}$, a RVM classifier can be trained to separate positive and negative samples in an iterative manner through maximising the marginal likelihood. Under the incremental learning scheme, each sample in the training set will be *tested sequentially* to determine whether it will be included in or excluded from the classification model. Decision boundary will be updated once sample vector is added to or removed from the model.
Multi-class Classification: The RVM classifier can be *extended* to a multi-class classifier by using the "one-versus-others" method. Since we have a total of 10 classes of motion, 10 independent RVM classifiers are constructed and each of them is trained to separate one class of data from all others. After all the classifiers are trained, the system can be tested by feeding a sample to all the classifiers. In practice, suppose this sample belongs to class i, the classifier which is trained to separate class i data from the others will give the largest output.
Re-training Procedure: If a new sample of class i is used to re-train the RVM classifiers, this sample will become a positive sample for $i-$th RVM classifier while become a negative sample for the other classifiers. All RVM classifiers will be trained separately by evaluating the relevance of this new sample. This sample will be either *added* to or *ignored* by the *classification model* of each RVM classifier. The *decision boundary* of each classifier will also be re-evaluated accordingly. In other words, if the new sample is quite different from the previously trained samples due to inter- and intra-personal variation, the classification model and the associated decision boundary will be adjusted to account for its influence. This implies adaptability can be achieved through the integration of a new training sample to the previously trained model.

4 Experiments

The proposed method was implemented using unoptimised C++ code and the OpenCV library. All the experiments described were executed on a P4 2.4GHz computer with 1G memory.

4.1 Experiments on synthetic data

We first utilise a set of synthetic data to illustrate how incremental training scheme improves the adaptability. An initial training set consisted of 2 classes, where class I (denoted by '$\times$' in a lighter intensity) was sampled from a mixture of 2 Gaussians while class II (denoted by '$\bullet$' in a lighter intensity) was sampled from a mixture of 3 Gaussians. Similarly, a training set for re-training the classifier consisted of 2 classes. Class I was still sampled from the same distribution as those used to generate the initial training set. Class II, however, was sampled from a mixture of 3 Gaussians whose means are *shifted upward* compare to the Gaussian mixture that generated the initial training set (see Figure 3). The sample points for re-training are shown in a darker intensity.

Firstly, the initial training set was used to train a RVM classifier using the incremental learning scheme. Afterwards, the re-training set was used to re-train the RVM classifier. The training results are illustrated in Figure 3. The plot shows that the decision boundary *adjusts (bends upwards) automatically* to separate the 're-training' samples from different classes. Another RVM classifier was trained by these training sets using batch learning. Similar decision boundary was achieved but the training time was longer (7344 ms vs. 766 ms).

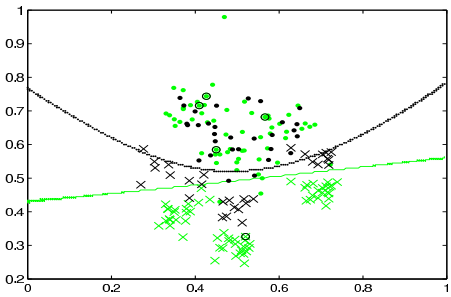

Fig. 3: *This figure illustrates the training result on the initial training set (denoted by lighter '×' and '•') by a RVM classifier using incremental learning and also the re-training result on the re-training set (denoted by darker markers) by the same RVM classifier. The decision boundary obtained from training using the initial training set is shown as a lighter line while the decision boundary obtained from training using the re-training set is shown as a darker line. Relevance vectors are shown circled.*

4.2 Experiments on real data

In this part, we will use video data to evaluate the performance of the RVM classifier using incremental learning. Both training and testing data were video captured under arbitrary room conditions (with various backgrounds and lighting). The video was captured by a webcam with a resolution of 320×240 pixels at 15 frames per second. In each video clip, the signer signs one of the ten primitive movements as described in Section 2. On average, each movement, which is manually segmented, lasts between 2 and 5 seconds.

We have five pairs of training set and testing set. Different pairs are captured under different conditions. Each dataset has a size of 300 (where each class of movement contributes to 30 samples). The first pair captured the motion of a signer (subject I) who signs the primitive movements using *hand shape 'B5'* (see Figure 4). The second, third and fourth pairs captured the motion of the same signer who signs using *hand shape 'B'*, to sign in a *faster speed*, and to sign with a slight *deviation in direction* respectively. The fifth pair captured the motion of *another signer* (subject II) who signs in the same way as subject I did in

the first pair. The difference in capturing conditions between these datasets and their corresponding MGO images generated are illustrated in Figure 4.

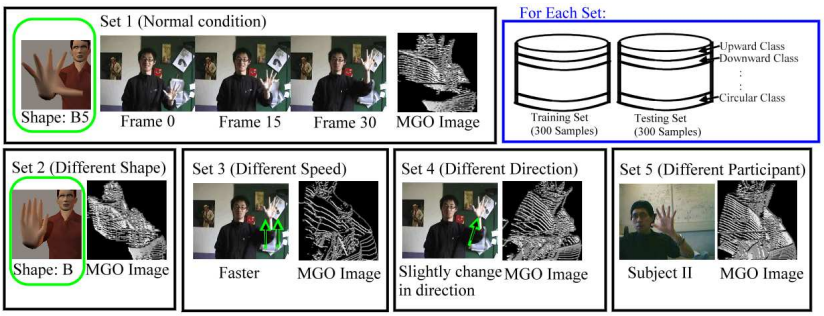

Fig. 4: *This figure illustrates the difference in capturing conditions between the datasets used and the corresponding MGO images produced. In all cases, the signers sign the 'Upward' gesture.*

The training set from the first pair was exploited to train a RVM classifier and a SVM classifier (both have the same kernel configuration). The testing sets of all five pairs of dataset were used to evaluate the performance of these classifiers. The training and testing results are summarised in Table 1.

Table 1: *This table shows the training and testing results on all testing sets by a SVM classifier and a RVM classifier.*

	SVM	incremental RVM
Training Time (ms)	2156	885906
Average No. of RVs	140	3
Classification Time (ms)	15.8	4.2
Accuracy on	Accuracy% (Unrecognised%)	
Set 1 (normal)	0.80 (0.18)	0.93 (0.03)
Set 2 (hand shape)	0.04 (0.96)	0.21 (0.68)
Set 3 (hand speed)	0.05 (0.95)	0.29 (0.57)
Set 4 (direction)	0.06 (0.93)	0.32 (0.53)
Set 5 (subject)	0.12 (0.87)	0.39 (0.59)

The training sets of the remaining four pairs (i.e. Set 2, 3, 4, and 5) were exploited to re-train the RVM classifier. We used *different amount of training data* per each pair to re-train the classifier (sample sizes used are 10, 20, 50, 100, 300). The testing sets of all five pairs were used to test the system. The training and testing results are shown in Table 2.

Table 2: *This table shows the training and testing results of using RVM classifiers that are re-trained by a different amount of training samples.*

No. of Re-training Samples per Training Set	10	20	50	100	300
Training Time (ms)	18922	27796	81344	140281	1085406
Average No. of RVs	4.8	6.0	7.4	9.2	12.5
Accuracy on	Accuracy% (Unrecognised%)				
Set 1 (normal)	0.90 (0.03)	0.92 (0.02)	0.90 (0.03)	0.90 (0.04)	0.91 (0.03)
Set 2 (hand shape)	0.35 (0.3)	0.72 (0.11)	0.88 (0.05)	0.93 (0.01)	0.94 (0.01)
Set 3 (hand speed)	0.54 (0.18)	0.66 (0.10)	0.82 (0.04)	0.91 (0.02)	0.93 (0.01)
Set 4 (direction)	0.47 (0.22)	0.63 (0.09)	0.78 (0.04)	0.86 (0.04)	0.89 (0.02)
Set 5 (subject)	0.54 (0.25)	0.82 (0.12)	0.91 (0.04)	0.92 (0.02)	0.92 (0.02)

4.3 Discussion

The experimental results illustrate three main advantages of using incremental RVM for motion recognition. Firstly, a RVM classifier maintains a *sparse model* and can thus perform classification with a *minimum amount* of *online* computational resources. Experiments show that the RVM classifier maintains a sparser model than the SVM classifier (3 RVs vs. 140 SVs). The time taken for performing RVM classification on a feature vector is shorter (4.2 ms by RVM vs. 15.8 ms by SVM). Since the time taken for extracting the motion features is 34.3 ms, the total time for performing RVM classification on video data is 38.5 ms (i.e. 26 frames per second). That is to say, the system can run in *real-time*.

Secondly, a RVM classifier returns a *probabilistic result*, which can provide information about uncertainty and facilitate high-level inference. Experiments demonstrate that the number of *unrecognisable samples* is higher if a SVM classifier is used. This is mainly because the RVM classifier has a better *generalisability* than the SVM classifier. Apart from this, the relatively poor classification result of SVM may also be due to the non-probabilistic nature of its output. Under the "one-versus-others" scheme, if all SVM classifiers give '0' response, the system will conclude that the input is not recognisable. In contrast, RVM classifiers give probabilistic values as output, the final decision will be made based on these values and will seldom give unrecognisable results with the exception that all probabilistic values are too low.

Thirdly, a RVM classifier allows incremental learning and thus has a higher *adaptability* to new samples captured under different environment. Experiments indicate that re-training a RVM classifier using a small amount of new samples is sufficient to achieve a fairly *high accuracy* when the classifier is applied on unseen data, which is captured under a different condition. In other words, re-training of a RVM classifier enables a better adaptability of the classifier towards variations, such as changes in hand shape, moving speed and moving direction, and even different person.

Experiments also reflect the main problem of using RVM classifiers is its relatively *long training time* compare with SVM classifiers. Incremental learning,

however, does shorten the learning time (7344 ms by batch learning vs. 766 ms by incremental learning). In addition, it is worth spending less than 2 minutes on re-training a RVM classifier with a small amount of new samples to achieve a better classification result.

5 Conclusion

A new method is proposed to increase the adaptability of a gesture recognition system that can be extended to sign language recognition. The proposed method performs better than recently used methods in three ways. Firstly, through the use of an incremental learning approach, newly available samples can be exploited to re-train the classifier that has been trained by an initial training set. Such an online re-training scheme can increase the *adaptability* of the classifier to input captured under a new condition. Secondly, by using a sparse Bayesian classifier that has relatively better *generalisability* and *sparsity*, the final classification result is comparable to other motion recognition methods and the result can be obtained with a minimum amount of online computational resources. Finally, the *probabilistic* nature of the Bayesian classifier implies that the proposed method can be applied in *complex motion* analysis that must maintain multiple hypotheses. A further investigation of how to extend this work to analyse complex motion and how to further reduce the *training time* is under progress.

Acknowledgements. SW is funded by the Croucher Foundation Scholarships.

References

1. Starner, T., Weaver, J., Pentland, A.: Real-time american sign language recognition using desk and wearable computer based video. PAMI **20** (1998) 1371–1375
2. Vogler, C., Metaxas, D.: A framework for recognizing the simultaneous aspects of american sign language. CVIU **81** (2001) 358–384
3. Bauer, B., Kraiss, K.F.: Video-based sign recognition using self-organizing subunits. In: Proc. ICPR. (2002) 282–296
4. Wilson, A., Bobick, A.: Realtime online adaptive gesture recognition. In: Proc. ICPR. (2000) Vol I: 270–275
5. Bowden, R., Windridge, D., Kadir, T., Zisserman, A., Brady, M.: A linguistic feature vector for the visual interpretation of sign language. In: Proc. ECCV. (2004) Vol I: 390–401
6. Derpanis, K.G., Wildes, R.P., Tsotsos, J.K.: Hand gesture recognition within a linguistics-based framework. In: Proc. ECCV. (2004) 282–296
7. Stokoe, W.C., Casterline, D., Croneberg, C.: A Dictionary of American Sign Language. Linstok Press, Washington, DC (1965)
8. Tipping, M.E.: Sparse bayesian learning and the relevance vector machine. The Journal of Machine Learning Research **1** (2001) 211–244
9. Tipping, M.E., Faul, A.C.: Fast marginal likelihood maximization for sparse bayesian models. In: Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics. (2003)
10. Bradski, G.R., Davis, J.W.: Motion segmentation and pose recognition with motion history gradients. Machine Vision and Applications **13** (2002) 174–184
11. Bobick, A.F., Davis, J.W.: The recognition of human movement using temporal templates. PAMI **23** (2001) 257–267