Sign Spotting using Hierarchical Sequential Patterns with Temporal Intervals

Eng-Jon Ong, Oscar Koller, Nicolas Pugeault, Richard Bowden
CVSSP, University of Surrey
Guildford GU27XH, UK
{e.ong,n.pugeault,r.bowden}@surrey.ac.uk, oscar.koller@gmail.com

Abstract

This paper tackles the problem of spotting a set of signs occuring in videos with sequences of signs. To achieve this, we propose to model the spatio-temporal signatures of a sign using an extension of sequential patterns that contain temporal intervals called Sequential Interval Patterns *(SIP). We then propose a novel multi-class classifier that organises different sequential interval patterns in a hierarchical tree structure called a Hierarchical SIP Tree (HSP-Tree). This allows one to exploit any subsequence sharing that exists between different SIPs of different classes. Multiple trees are then combined together into a forest of HSP-Trees resulting in a strong classifier that can be used to spot signs. We then show how the HSP-Forest can be used to spot sequences of signs that occur in an input video. We have evaluated the method on both concatenated sequences of isolated signs and continuous sign sequences. We also show that the proposed method is superior in robustness and accuracy to a state of the art sign recogniser when applied to spotting a sequence of signs.*

1. Introduction

Automated Sign Language Recognition (SLR) remains a challenging problem to this day. Like spoken languages, sign language feature thousands of signs, sometimes only differing by subtle changes in hand motion, shape or position. This, compounded with differences in signing style and physiology between individuals, makes SLR an intricate challenge.

A large body of work on automated SLR has focused on isolated signs, where a sequence contains only a single sign. Approaches can be divided between tracking-based, sub-unit classifiers [5], and more data driven approaches. As examples of the latter, Wang *et al*. [11], created an American Sign Language (ASL) dictionary based on similarity between signs using a Dynamic Space-Time Warping (DSTW) approach and Gavrilov *et al*. [4] proposed a data mining approach for detecting reduplications in signs. One prevalent family of methods for SLR are Hidden Markov Models (HMMs). Pitsikalis *et al*. [9] proposed a method which uses linguistic labelling to split signs into sub-units. From this they learn signer specific models, which are then combined via HMMs to create a classifier. Two drawbacks of HMMs for sign recognition are that they are learnt in a non-discriminative manner and do not perform feature selection, resulting in sub-optimal classifiers for datasets with ambiguous classes (as is typical in SLR).

To overcome this, an alternative approach using discriminative spatio-temporal patterns, called Sequential Patterns (SPs) was proposed by Elliott *et al*. [2]. SPs are ordered sequences of feature subsets that allow for explicit spatio-temporal feature selection and do not require DTW for temporal alignment. Here SPs were learnt in a discriminatory fashion as 1vs1 classifiers before being combined into strong classifiers for recognising signs—hence the approach scaled poorly to large numbers of signs. To address this, Ong et al [8], proposed a method for building SP forests where multiple SPs that share initial subsequences are combined into a tree structure, producing an inherently multi-class classifier. This was then applied to isolated sign recognition and shown to outperform HMMs based techniques, yielding state-of-the-art recognition performance on a database with a large number of signs.

These articles addressed the recognition of *isolated* signs; in contrast, we focus on the more difficult problem of spotting and recognising a sequence of known signs in a video, without indications of start and end points. A number of previous studies approached similar problems using variants of HMMs. For example, Liang & Ouhyoung [6] presented a recognition system for Taiwanese sign language based on HMMs. Using data-gloves for extracting the features used for recognition, their system could classify 250 signs with 80% performance in a signer dependent experiment. Also, Elmezain [3] used HMMs for recognising 10 signs in unsegmented sequences, and Morency *et al*. [7] proposed a discriminative approach for unsegmented gesture detection based on Latent Dynamic CRF. However, the above gesture recognition work used only a limited set of

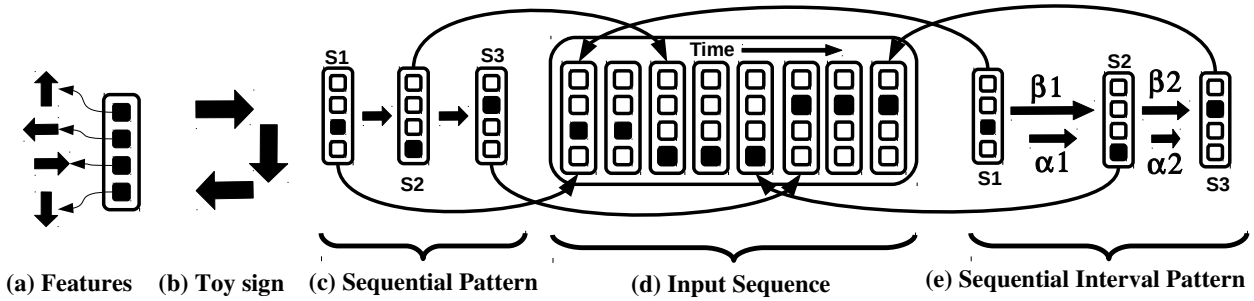


Figure 1: Illustration of SP and SIP encoding of a toy sign: (a) Toy binary feature vector encoding 4 movements of one hand; (b) Example trajectory for a toy sign; (c) Example of a SP describing (b): $\mathbf{S} = \langle S1, S2, S3\rangle$; (d) Example sequence containing the sign in (b); (e) Example of a SIP describing (b): $\mathbf{S}^* = \langle\langle S1, S2, S3\rangle, ((\alpha 1, \beta 1), (\alpha 2, \beta 2))\rangle$.

simple, distinctive gestures, whereas sign language recognition involves a large vocabulary of complex, and often ambiguous, signs. The work of Yang *et al*. [12] proposed a CRF-based approach for sign spotting with a lexicon of 48 signs but for a single signer.

This article presents an extension of the SP-Trees approach [8] that allow for efficent spotting of unsegmented signs in videos. To this end, we evaluate the spotting performance of the proposed method on concatenated isolated signs, as well as continuous sign sentences.The main contributions of this paper are: 1) a new extension to SPs incorporating temporal intervals (sec. 2); 2) a novel hierarchical tree structure called *Hierarchical Sequential Pattern Trees (HSP-Trees)* (sec. 3), that can exploit any subsequence shared between different signs for classification (sec. 4);. and 3) an efficient algorithm for learning HSP-Trees (sec. 5). We show experimentally that the proposed method results in significant improvements in accuracy over the state-of-the-art SP-Trees and HMMs for sign spotting (sec. 6). Finally, we conclude with future work in sec. 7.

2. Representing Signs using Sequential Patterns with Temporal Intervals

The aim of this article is to spot occurences of a lexicon of C signs in an N-frames video. All frames in the video are assigned a label $\mathcal{L} = \{\lambda_0, \lambda_1, \ldots, \lambda_C\}$, where λ_0 denotes the absence of any known sign (as per the lexicon). In order to describe signs, we extract a vocabulary of D static and dynamic features at every frame in the video (Fig. 1a) to form a vocabulary of features $\mathcal{E} = \{e_1, \ldots, e_D, \bar{e_1}, \ldots, \bar{e_D}\}$, where e_i and $\bar{e_i}$ represent the presence and absence of feature i, respectively. Therefore, a frame $i \in [1, N]$ in the video is coded by the itemset $S_i \subset \mathcal{E}$ and an N-frames video sequence by $\mathbf{S} = \langle S_1, \ldots, S_N\rangle$. We first introduce a formalism to represent the dynamic signature characterising signs as *Sequential Patterns*.

2.1. Sequential Patterns S

We call any sequence of itemsets a Sequential Pattern (SP). SPs can serve to encode either: 1) the sequence of features extracted from a video from the corpus; 2) a spatio-temporal signature unique to a particular sign label. Intuitively, a sequential pattern encodes the occurence of various feature sets in a specific order—see Fig. 1c. Importantly, SPs do not constrain what happens between two following feature sets, just that they need to occur in the specified order. Formally, we have:

Definition 1. Let $\mathcal{E} = \{e_1, \ldots, e_{|\mathcal{E}|}\}$ be a set of binary features, and $S \subset \mathcal{E}$ an *itemset*, then a *Sequential Pattern* $\mathbf{S} \in \mathcal{S}_\mathcal{E}$ of length L is defined as a sequence of itemsets $\mathbf{S} = \langle S_1, \ldots, S_L\rangle$.

In order for SPs to be used for recognition, we need to define a relation assessing whether a SP is a subsequence of another. For example, in Fig. 1, all itemsets in the left-hand SP (Fig. 1c) occur in the central sequence, and in the same order, hence we say that the left SP is a subsequence of the central one (Fig. 1d). More formally:

Definition 2. Let $\mathbf{A} = \langle A_i\rangle_{i=1}^{|\mathbf{A}|}$ and $\mathbf{B} = \langle B_i\rangle_{i=1}^{|\mathbf{B}|}$ be two SPs, we define that $\mathbf{A} \sqsubseteq \mathbf{B}$ iff. there exists a sequence $K = (k_1, \ldots, k_{|A|})$ such that $k_i < k_j, \forall i < j$ for which $\forall i \in \{1, \ldots, |A|\}$ we have $A_i \subseteq B_{k_i}$. We shall denote K as a *detection index set*.

Assuming we have two SPs such that $\mathbf{A} \sqsubseteq \mathbf{B}$, it will be common that A occurs in several different configurations in B (ie, the previous definition holds true for multiple index sets K). For this reason, we identify the *earliest* and *latest* occurences of A in B. Formally, the detection index set $K = (k_i)_{i=1}^{|\mathbf{A}|}$ is called the *earliest detection index set* if $k_1 = \text{argmin}_{i\in[1,|\mathbf{B}|]} A_1 \subset B_i$ and $k_i = \text{argmin}_{j\in[k_{i-1},|\mathbf{B}|} A_i \subset B_j, \forall i \in [2, |\mathbf{A}|]$. Similarly, the index set $K = (k_i)_{i=1}^{|\mathbf{A}|}$ is called the *latest detec-*

tion index set if $k_{|\mathbf{A}|} = \text{argmax}_{i\in[1,|\mathbf{B}|]} A_{|\mathbf{A}|} \subset B_i$ and $k_i = \text{argmax}_{j\in[1,k_{i+1}]} A_i \subset B_j$.

Note that SPs do not encode the time (or number of frames) elapsed between itemsets. This means that the likelihood for an SP $\mathbf{S}$ to be included by chance in a sequence $\mathbf{D}$ increases with the sequence's length. This makes SPs adequate for the recognition of temporally segmented and isolated signs, but weak for spotting signs in long video streams.

2.2. Sequential Interval Patterns $\mathbf{S}^*$

In order to overcome the weakness noted above, we propose an extension of the SP called Sequential Interval Pattern (SIP), which restricts the permitted temporal intervals between consecutive itemsets (see Fig. 1e).

Definition 3. A *Sequential Interval Pattern* $\mathbf{S}^* \in \mathcal{S}^*_\mathcal{E}$ of length L is defined as the tuple: $\mathbf{S}^* = \left\langle \langle S_i \rangle_{i=1}^{L}, (\alpha_i, \beta_i)_{i=1}^{L-1} \right\rangle$, where $S_i \subset \mathcal{E}$ is an itemset, and $\alpha_i, \beta_i \in \mathbb{R}$, $\alpha_i \geq \beta_i, \forall i \in \{1, \ldots, |\mathbf{S}^*|\}$, denotes the minimum and maximum duration allowed between itemsets S_i and S_{i+1}, respectively.

In this definition, each itemset S_i is assigned a tuple (α_i, β_i) that constrain the minimum and maximum interval (ie, number of frames) allowed between this itemset and the next. This is illustrated in Fig. 1, where the right SIP (Fig. 1e) is a subsequence of the central sequence (Fig. 1d). Although the itemsets in the right SIP are the identical to the left SP, fewer occurences in the central sequence are valid due to the restrictions provided by the (α_i, β_i) on the interval between itemsets. Formally, the subsequence relation in Def. 2 is rewritten:

Definition 4. We say that a SIP $A = \left\langle \langle I_i^A \rangle_{i=1}^{L_A}, (\alpha_i^A, \beta_i^A)_{i=1}^{L_A-1} \right\rangle$ of length L_A is included in another SIP $B = \left\langle \langle I_i^B \rangle_{i=1}^{L_B}, (\alpha_i^B, \beta_i^B)_{i=1}^{L_B-1} \right\rangle$ of length L_B, denoted by $A \sqsubseteq B$, iff. $\exists \{k_1, \ldots, k_{|A|}\}$, such that

$$\begin{cases} I_i^a \subseteq I_{k_i}^B \\ \alpha_i^A \leq \sum_{j=k_i}^{k_{i+1}-1} \alpha_j^B \\ \beta_i^A \geq \sum_{j=k_i}^{k_{i+1}-1} \beta_j^B \end{cases}$$

Note that SPs are a specific version of SPs, with the minimum interval set to 0 and maximum to infinity.

3. Hierarchical Sequential interval Pattern Trees (HSP-Trees)

This section presents an efficient tree structure for encoding, detecting and learning SIPs, called a Hierarchical Sequential Pattern Tree (HSP-Tree), which allow sharing of sub-patterns between multiple SIPs (see Fig. 2).

3.1. Definition of HSP-Trees

Formally, an HSP-Tree is a tuple $T = (N, L)$, where $N = \{n_1, \ldots, n_{|N|}\}$ is the set of nodes and $L = \{l_1, \ldots, l_{|L|}\}$ the set of links.

In an HSP-Tree, nodes $n \in N$ encode an SIP $\mathbf{S}^*$, and are assigned a label $\lambda \in \mathcal{L}$ corresponding to a sign in the lexicon (or the 'reject' label). Formally, a HSP-Tree node is defined as the tuple $n = (\mathbf{S}^*_n, \lambda_n, \bar{\alpha}_n, \bar{\beta}_n)$. There, $\bar{\alpha}, \bar{\beta}$ determine the maximal and minimal possible durations of the SIP $\mathbf{S}^*$ coded by the node. They are aggregated from the maximal and minimal intervals between all itemsets in $\mathbf{S}^*$, such that $\bar{\alpha}_n := \sum_{i=1}^{|\mathbf{S}^*_n|} \alpha_i$, and $\bar{\beta}_n := \sum_{i=1}^{|\mathbf{S}^*_n|} \beta_i$. For convenience, we use a functional notation to denote nodes' properties: $\widehat{\mathbf{S}^*}[n] := \mathbf{S}^*_n$, $\widehat{\lambda}[n] := \lambda_n$, $\widehat{\alpha}[n] := \bar{\alpha}_n$ and $\widehat{\beta}[n] := \bar{\beta}_n$.

The links are oriented, and ensure that the nodes are connected in a tree structure as well as providing information on how to traverse the tree. Formally, a link $l \in L$ is the tuple $l = (n, n', t)$, such that $t \in \{+, -\}$ is the link type and $n, n' \in N$ are two nodes, called the *parent* and *child node*, respectively, such that $n \neq n'$. The link type t is used to define how the tree is traversed at learning and detection time—see sections 4 and 5.

For convenience we define the descendance $d(n) \subset N$ of a node $n \in N$ as the subtree starting from this node:

Definition 5. We say that $n' \in N$ is a descendant of $n \in N$ iff. $\exists \{k_1, \ldots, k_i\}$ and $\{s_1, \ldots, s_{i-1}\}$, such that $n_{k_1} = n$, $n_{k_i} = n'$, $(n_{k_j}, n_{k_{j+1}}, s_j) \in L, \forall j \in [1, i-1]$. We denote the set of all descendants of n as $d(n)$.

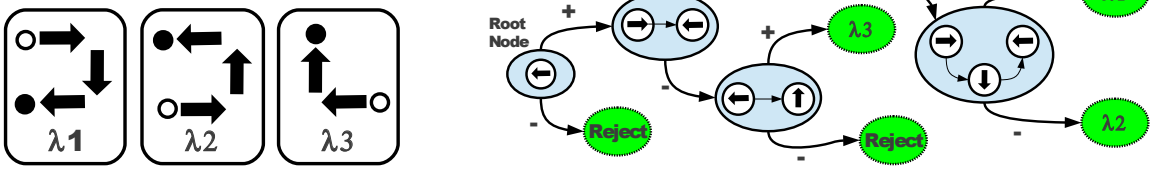
Additionally, a tree $T = (N, L)$ has a unique *root node* $r(T) \in N$ that is the descendant of none, and a set of nodes called *leaf nodes* $L(T) \subseteq N$ that have no descendant.

For a tree to be an HSP-Tree, two constraints need to be satisfied: First, HSP-Trees are binary, meaning that all non-leaf nodes have exactly one positive and one negative descendants:

Property 1. *Let $T = (N, L)$ be a HSP-Tree. Then for each non-leaf node $n \in N$, it will only have two child nodes: n^+ and n^-, $n^+ \neq n^- \neq n$, where $(n, n^+, +) \in L$ and $(n, n^-, -) \in L$, called the* positive *and* negative *child of n respectively. For convenience, we define the following accessor functions: $+[n] = n^+$ and $-[n] = n^-$.*

Second, the SIP encoded in a non-leaf node is a subsequence of the SIP encoded by its positive child, and all the descendants thereof (Fig. 2b). This property provides the crucial mechanism that allows us to merge different SIPs with shared subsequences:

Property 2. *Let $T = (N, L)$ be a HSP-Tree, and $n \in N$ a node, then if $(n, n', +) \in L$, then we have $\widehat{\mathbf{S}^*}[n] \sqsubseteq \widehat{\mathbf{S}^*}[n'']$, $\forall n'' \in d(n') \cup \{n'\}$.*



(a) Example of 3 toy signs, starting at the white circle and ending at the black circle.

(b) Correponding HSP-Tree for classifying the toy signs.

Figure 2: Illustration of how the HSP-Tree combines sequential patterns of three toy signs (a) to form a multi-class classifier (b). Non-leaf nodes are shown in blue with its associated SIP. For clarity, the SIP temporal intervals are not shown. Leaf nodes are shown in green. The link types are denoted by the +,- signs. Importantly, the HSP-Tree allows two SPs with any common subsequences (which may be in the middle of both SPs) to be merged. This type of sharing would not be possible with the SP-Tree method.

Hence, we can say that the positive descendence of a node n *specializes* the SIP $\widehat{\mathbf{S}^*}[n]$—this is exemplified in Fig. 2b.

4. Classification using HSP-Forests

In this section we describe how an HSP-Tree can be used to assign labels to frames in a video for sign spotting. First, sec. 4.1 describes the classification of segmented videos containing a single sign; then sec. 4.2 provides an extension to label an unsegmented video sequence containing multiple signs.

4.1. Individual Sign Classification

The classification of a sequence containing only a single sign is a function $\phi : \mathcal{T}_\mathcal{E} \times \mathcal{D}_\mathcal{E} \rightarrow \mathcal{L}$ that takes in an input sequence and an HSP-Tree ($T = (N, L)$) and outputs a label.

In practice, the classification will be performed by starting at the root node $r(T)$, and traversing the tree until a leaf node $n \in L(T)$ is reached. The tree is traversed by following the positive link out of a node if the SIP it encodes is a subsequence of $\mathbf{D}$, and following the negative link otherwise. Formally, we define the function $\zeta : N \times \mathcal{D}_\mathcal{E} \rightarrow N$ that traverses the tree for an input sequence $\mathbf{D} \in \mathcal{D}_\mathcal{E}$, and returns a leaf node:

$$\zeta(n, \mathbf{D}) = \begin{cases} n & \text{if } n \in L(T) \\ \zeta(+[n], \mathbf{D}) & \text{if } \mathbf{S}^*[n] \sqsubseteq \mathbf{D} \\ \zeta(-[n], \mathbf{D}) & \text{otherwise} \end{cases} \quad (1)$$

When a leaf node is reached, the sequence is assigned this node's label. In sum, an input sequence $\mathbf{D} \in \mathcal{D}_\mathcal{E}$ is classified by an HSP-Tree T as:

$$\phi(T, \mathbf{D}) = \widehat{\lambda}[\zeta(r(T), \mathbf{D})] \quad (2)$$

Finally, if we have a population of K weighted HSP-Trees $\mathbf{F} = \{(T_1, \alpha_1), \ldots, (T_K, \alpha_K)\}$, called in the following an HSP-Forest, the sequence $\mathbf{D}$ is classifed as the weighted majority vote:

$$\Phi(\mathbf{F}, \mathbf{D}) = \underset{\lambda \in \mathcal{L}}{\operatorname{argmax}} \sum_{(T_j, \alpha_j) \in \mathbf{F}} \alpha_j \mathbb{I}[\phi(T_j, \mathbf{D}) = \lambda] \quad (3)$$

where $\mathbb{I}[x]$ is an indicator function and the weights α_j are determined by the boosting framework—see sec. 5.2.

4.2. Sequence of Signs Classification

The previous approach is suitable for assigning a *unique* label to a segmented sequence that contains a single sign. For the more general problem of *sign spotting*, we have a long video sequence containing multiple signs and no information about when each sign starts and finishes, and we need to assign a label to each frame in the sequence.

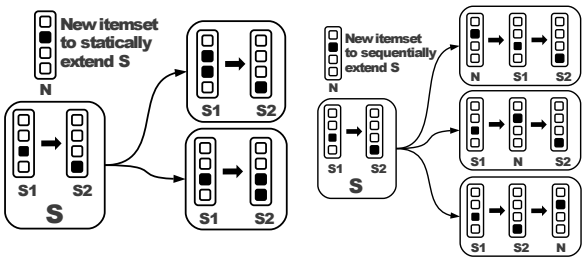
We propose to extend the classification method in the previous section with a temporal scanning window, starting at the frame to be classified. Formally, let $\mathbf{D} \in \mathcal{D}_\mathcal{E}$ be a (long) data stream, we define a scanning window on $\mathbf{D}$ as a function that generate a sub-stream $W : \mathcal{D}_\mathcal{E} \times \mathbb{N} \rightarrow \mathcal{D}_\mathcal{E}$ of length Q, such that:

$$W_Q(\mathbf{D}, i) = \langle D_j \rangle_{j=i}^{i+Q} \quad (4)$$

where $D_j \subset S_\mathcal{E}$ are itemsets on $\mathcal{E}$. It follows from this definition of the scanning window that any frame $i \in [1..|\mathbf{D}| - Q]$ in the sequence will be contained in exactly Q windows.

Each frame i in the sequence $\mathbf{D}$ is then assigned a label $\lambda \in \mathcal{L}$ as the majority vote of the HSP-Forest over all Q-windows that contain this frame:

$$\Phi^Q(\mathbf{F}, \mathbf{D}, i) = \underset{\lambda \in \mathcal{L}}{\operatorname{argmax}} \sum_{k=i-Q}^{i} \mathbb{I}[\Phi(\mathbf{F}, W_Q(\mathbf{D}, k)) = \lambda], \quad (5)$$



(a) Static extension of S. (b) Sequential extension of S.

Figure 3: Methods for extending a SP or SIP.

5. Learning the HSP Forest

In this section, we describe the method for learning a set of HSP-Trees that will be linearly combined together into a strong classifier within the Boosting framework. Let us assume a training set $\mathbf{X} \subset \mathcal{D}_{\mathcal{E}}$, with N training examples: $\mathbf{X} = \{X_i\}_{i=1}^{N}$, where $X_i \in \mathcal{D}_{\mathcal{E}}$ is a sequence of D-dimensional binary feature vectors ($|\mathcal{E}| = D$), features which will be detailed in sec. 6. The length of the sequence X_i is denoted as $|X_i|$ and we define $X_i = (x_i)_{i=1}^{|X_i|}$, $x_i \in \{0,1\}^D$. Associated with the sequences in $\mathbf{X}$, we have a set of labels $Y = (y_i)_{i=1}^{N}$, $y_i \in \mathcal{L}$ and a normalised set of example weights $W = (w_i)_{i=1}^{N}$, where $\sum_{i=1}^{X} w_i = 1$. For convenience, the weight of an example X is denoted as $w(X)$ and its label as $y(X)$. We also define the function $\lambda : \mathbf{X} \rightarrow \mathcal{L}$, that gives the most frequent label in a set of examples $\mathbf{X}$:

$$\lambda(\mathbf{X}) = \underset{c \in \mathcal{L}}{\operatorname{argmax}} \sum_{X \in \mathbf{X}} \mathbb{I}\left[y(X) = c\right] \tag{6}$$

5.1. Building the HSP-Tree

HSP-Trees are built in a manner similar to typical decision trees, where the input $\mathbf{X}$ is recursively split between sequences that contain a given SIP $\mathbf{X}^+ = \{X \in \mathbf{X} | S \sqsubseteq X\}$, according to Def.4, and the others $\mathbf{X}^- = \{X \in \mathbf{X} | S \not\sqsubseteq X\}$. The learning algorithm finds the SIP S that provides the split of the training set with optimal Gini criteria.

For efficiency, the optimal SIP is learnt in two steps: the first step finds the optimal SP irrespective of intervals (sec. 5.1.1), the second optimises SIP intervals from the training data (sec. 5.1.2).

5.1.1 Optimally Extending SIPs

This algorithm sequentially extends existing a (possibly empty) SIP in order to maximise the Gini index. We define two ways of extending a SIP: 1) adding new items to an existing itemset (*static extension*); 2) adding a new itemset before, between or after its itemsets (*sequential extension*).

Static extension: Let $S = \left\langle \langle S_i \rangle_{i=1}^{L}, (\alpha_i, \beta_i)_{i=1}^{L-1} \right\rangle$ be an SIP of length L, $t \in [1, L]$ be the itemset index of S to extend with the new item $d \in [1, D]$. We can produce a *static extension* of S at itemset t called $S' = \left\langle \langle S'_i \rangle_{i=1}^{L}, (\alpha_i, \beta_i)_{i=1}^{L-1} \right\rangle$ where:

$$S'_i = \begin{cases} S_i & \text{if } i \neq t \\ S_i \cup \{d\} & \text{otherwise} \end{cases} \tag{7}$$

This is shown in Fig. 3a. This operation of a static extension is denoted using the $\cup_t^{st}$ operator: $S' = S \cup_t^{st} d$.

Sequential extension: Let $S = \left\langle \langle S_i \rangle_{i=1}^{L}, (\alpha_i, \beta_i)_{i=1}^{L-1} \right\rangle$ be an SIP of length L, and $\{d\}, d \in [1, D]$ be a new itemset to insert in S at a location $t \in [0, L]$. We can produce a *sequential extension* of S, defined as $S' = \left\langle \langle S'_i \rangle_{i=1}^{L+1}, (\alpha'_i, \beta'_i)_{i=1}^{L} \right\rangle$ such that:

$$\langle S'_i \rangle_{i=1}^{L+1} = \begin{cases} (\{d\}, S_1, ..., S_L) & \text{if } t = 0 \\ (S_1, ..., S_t, \{d\}, S_{t+1}, ..., S_L) & \text{if } 0 < t < L \\ (S_1, ..., S_L, \{d\}) & \text{otherwise} \end{cases} \tag{8}$$

This is shown in Fig. 3b. This operation is denoted by the $\cup_t^{se}$ operator, with $S' = S \cup_t^{se} d$.

The proposed learning algorithm makes use of these two extension operators $\cup_t^{st}$ and $\cup_t^{se}$ to find the optimal extension $S' = S \cup_{t'}^{s'} d'$ of the existing SIP S, according to the criterion γ, such that:

$$d', t', s' = \underset{d \in [1,D], t \in [0,L], s \in \{st, se\}}{\operatorname{argmin}} \gamma(S \cup_t^s d, \mathbf{X}) \tag{9}$$

Note that temporal interval values are ignored at this stage, and will be optimised in the next step.

5.1.2 Configuring Interval Values

The next step requires an optimisation of the interval values $(\alpha_i, \beta_i)_{i=1}^{L-1}$ of the SIP S. First, we determine the minimum and maximum intervals between an SIP's itemsets for a single sequence:

Theorem 1. *Let $X = (X_i)_{i=1}^{N}$ be an example sequence of length N and $S = \left\langle \langle S_i \rangle_{i=1}^{L}, (\alpha_i, \beta_i)_{i=1}^{L-1} \right\rangle$ be a length L SIP. Let $K' = (k'_i)_{i=1}^{L}$ and $K'' = (k''_i)_{i=1}^{L}$ be the earliest and latest detection index set of $(S_i)_{i=1}^{L}$ respectively. Then, the maximum possible interval value between itemsets S_i and S_{i+1} is $\Delta_i = k''_{i+1} - k'_i$.*

Given an SIP $S = \left\langle \langle S_i \rangle_{i=1}^{L}, (\alpha_i, \beta_i)_{i=1}^{L-1} \right\rangle$ and a set of training examples $X = (X_i)_{i=1}^{N}$, Theorem 1 provides the maximum intervals for each pair of consecutive itemsets in S for each training example X_j. If we denote the maximum

interval value between S_i and S_{i+1} in example X_j as Δ_i^j, then, our search range for the intervals (α_i, β_i) is given by:

$$\sigma_i = [0, \max_{j \in [1,N]} (\Delta_i^j)] \quad (10)$$

The optimisation of the intervals $(\alpha_i, \beta_i)_{i=1}^{L-1}$ of S is described in Algo. 1: First, all intervals are initialised to $(1, \infty)$ such that S is equivalent to an SP.Then, the optimal values for each interval (α_i, β_i) are determined sequentially whilst holding all other interval values constant.

Algorithm 1 Sequential Interval Optimisation Algorithm

Input: Training Examples $\mathbf{X} = (X_i)_{i=1}^N$
Input: a SP: $S = \langle S_i \rangle_{i=1}^L$
Output: a SIP: $S^* = \langle S, (\alpha_i, \beta_i)_{i=1}^{L-1} \rangle$.
Initialise intervals of S^*: $\forall i \in [1, L-1], \alpha_i = 1, \beta_i = \infty$
for $i \in [1, L-1]$ **do**
  Let γ_i' be the Gini impurity with $\alpha_i = \alpha'$ and $\beta_i = \beta'$.
  Let σ_i (Eq. 10) be the search range for the i^{th} interval.
  Set $(\alpha_i, \beta_i) \leftarrow \text{argmin}_{\alpha' \in \sigma_i, \beta' \in \sigma_i} \gamma_i'$
end for
Return S^*

5.1.3 HSP-Tree Building Algorithm

Using the tools provided in sec. 5.1.1 and 5.1.2, we describe the HSP-Tree learning algorithm in Algo. 2. At each non-leaf node, the dataset is split according to the node's SIP S', that is extended from its closest *positive* ancestor's SIP S (possibly being the empty SIP if the node is connected to the root by negative links only), in such a way as to greedily optimise the Gini criterion, as explained in sec. 5.1.1 and 5.1.2. Both subsets $\mathbf{X}^+$ and $\mathbf{X}^-$ resulting from this split are then sent to the positive and negative children of the node (respectively). The positive child node will then further extend S' on the subset $\mathbf{X}^+$ and the negative child extends S on the subset $\mathbf{X}^-$, splitting recursively the input space with more specific SIPs.

This process continues until one of 3 termination criteria is met: 1) maximum tree-depth O_{max} is reached; 2) training subset is smaller than minimum size N_{min} (set here as 1); or 3) the training subset is "pure" (i.e., contains samples that all belong to the same class).

5.2. Boosting HSP-Forests

In order to build the HSP-Forest, a modified version of the multi-class AdaBoost method is used. Here, the weak learner selection procedure is replaced by Algo. 2 for iteratively building appropriate HSP-Trees, given a set of weighted and labelled examples.

Algorithm 2 HSP-Tree Learn Algorithm

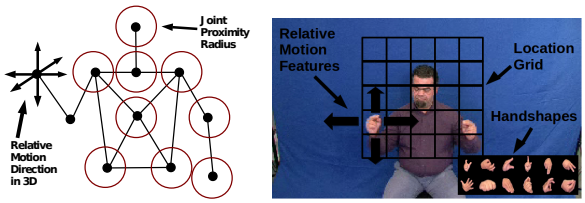
Input: Training Set: $(\mathbf{X}, Y, W)$
Output: HSP-Tree $\mathbf{T}$
Queue element: (Node, Train Subset, Depth)
Initialise empty SIP for root node: $S_I = \langle \langle \rangle, () \rangle$.
Set root node $R = (S_I, \lambda(\mathbf{X}))$
Initial HSP-Tree Nodes: $\mathbf{N} = \{R\}$
Initial HSP-Tree Edges: $\mathbf{E} = \emptyset$
Initialise queue: $Q = \{(R, \mathbf{X}, 1)\}$.
while $Q \neq \emptyset$ **do**
  Remove last item of Q: $(N^{cur}, \mathbf{X}^{cur}, O^{cur})$
  Let $S^{cur} = \widehat{\mathbf{S}^*}[N^{cur}]$
  if $|\mathbf{X}^{cur}| \leq N_{min}$ or $O^{cur} \geq O_{max}$ or $\gamma(S^{cur}, \mathbf{X}^{cur}) = 0$ **then**
    Update node label: $N^{cur} \leftarrow (S^{cur}, \lambda(\mathbf{X}^{cur}))$
  else
    Update N^{cur} with optimal extension of its SIP:
      Extend S^{cur} using Eq. 9, giving new SIP: S'.
      Configure the intervals of S' using Algo. 1
      Update node content: $N^{cur} \leftarrow (S', \lambda(\mathbf{X}^{cur}))$
      Partitions of N^{cur} : $\mathbf{X}_{cur}^+$ and $\mathbf{X}_{cur}^-$.
    Update the queue:
      New nodes: $L = (S^{cur}, -1)$ and $K = (S', -1)$
      $\mathbf{N} = \mathbf{N} \cup \{L, K\}$.
      $\mathbf{E} = \mathbf{E} \cup \{(N^{cur}, L, -), (N^{cur}, K, +)\}$.
      $Q = Q \cup \{(K, \mathbf{X}_{cur}^+, O^{cur}+1), (L, \mathbf{X}_{cur}^-, O^{cur}+1)\}$.
  end if
end while
Return HSP-Tree: $\mathbf{T} = (\mathbf{N}, \mathbf{E})$

6. Experiments

The experiments in this section aim to evaluate the performance of the proposed method for spotting individual signs in sequences. To this end, three publically available databases are used: a 40-sign dataset with multiple subjects [8], allowing us to evaluate accuracy in subject dependent and independent settings; a large 981-sign dataset evaluating the scalability of our approach [8] and finally a database for spotting signs within continuous sign language sentences [1], allowing us to evaluate how well co-articulation effects are handled.

6.1. Databases and Features

The first dataset (Dataset I) is based on 40 German Sign Language (DGS) signs; a mixture of both similar and dissimilar signs. Data from 14 subjects were captured using a Kinect camera, with 5 repetitions per sign. None of the subjects were native signers, leading to large variations in signing styles. The resulting dataset is challenging, featuring both inter and intra signer differences. In this dataset, the 24D binary feature vector in [8] was used (Fig. 4a), which consists of: 1) 6 features encoding the relative 3D movement of each hand; 2) Positional proximity to 9 joints;



(a) Dataset I Features (b) Dataset II Features

Figure 4: Features used for both datasets

3) dual hand features of both hands comming together, moving apart and synchronised movements.

The second dataset (Dataset II) contains videos of a native signer performing 981 Greek Sign Language (GSL) signs (a mixture of similar and dissimilar signs) with 5 repetitions. The 65D binary feature vector (Fig. 4b) from [8] was used in this dataset, which consists of: 1) 8 features of 2D relative motion extracted from tracked hand trajectories; 2) 4 dual handed bimanual features encoding both hands moving together, apart and in synchrony; 3) 40 features of the hand location on a grid placed relative to the head; 4) 12 features for representing different hand-shapes based on HOG features classified using decision forests.

The third dataset (Dataset III) contains 201 videos, each containing a native signer performing a continuous sign language sentence along with tracked hands and head positions. This dataset is challenging due to co-articulation that is present across similar signs. Additionally, unlike HMMs, we do not make use of any grammar model, as this work is concerned with sign spotting solely. Although there is a total of 104 signs, most occur only few times in the dataset, making training and testing difficult. Thus, a subset of signs with at least 3 training instances was chosen, resulting in a total of 48 signs. The features used are similar to the relative motion features and hand shape features used in Database II.

6.2. Experimental Setup

For experiments on Dataset I and II, the original dataset was split into two partitions: the training set and test set. A multi-sign test sequence is produced by concatenating single sign sequences ordered by a random permutation sequence of all the sign labels. The label permutation sequence will also act as the *groundtruth sequence* for this multi-sign sequence.

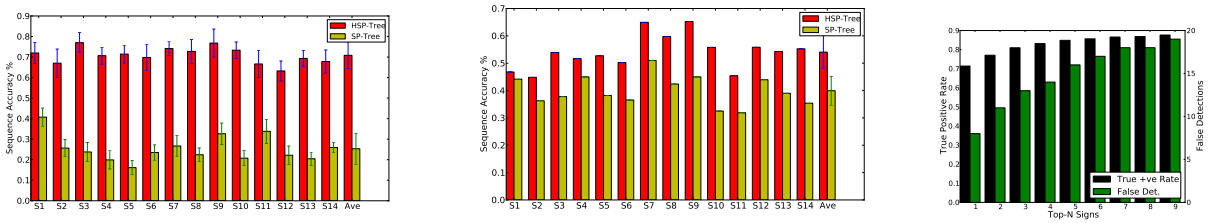
For Dataset I, we partition the dataset in two ways: 1) Leave-one-out signer dependent, where an example from each sign and subject is removed and assigned to the test set; 2) Cross-validated signer independent, where all examples from a subject are assigned to the test set. For Dataset II, we partition in the same manner as the signer dependent set in Dataset I, but evaluate the performance at different number of signs ranging from 100 to 981. For both Dataset I and II, the accuracy is evaluated similarly to word accuracy in HMMs. In Dataset III, we evaluate the results based on the percentage of true positives and number of false positives of signs from the output sequence compared with the ground truth labels. For all the experiments, 200-tree HSP-Forests were trained with maximum depth of 20 and minimum examples of 2. For comparisons against state-of-the-art, SP-Tree-based classifiers were also trained using the same settings.

Additionally, we also compare against the performance of HMMs, using the open-source speech recognition system RASR [10]. The HMMs are trained using single Gaussian densities in Bakis structure with two consecutive states sharing the same distribution. The number of states reflects the actual length of the training sequences. The binary features are PCA reduced to maintain 99% of their original variance and a 0-gram language model for equal likelihood of all classes is used to simulate the lack of a language model.

6.3. Experimental Results

The subject dependent results on Database I can be seen in Fig. 5a. It can be seen that the SP-Tree method performs poorly when the signs boundaries are not known, which is the case for a sequence of signs. In contrast, the performance of the proposed HSP-Trees are consistently better, by a significant amount, giving an average accuracy of 71.1% vs 26.9%. In comparison, the performance of HMMs in this experiment was 63.0%. In the subject independent tests (Fig. 5b), we again see that the proposed method is consistently better than the SP-Trees, with an accuracy of 54.1% vs 39.9% (HMMs: 49.3%). For the experiments on Dataset II, HSP-Trees again consistently outperform SP-Trees, with an average of 35% more accurate than SP-Trees. This divergence in accuracy is present even when the number of signs are large (981 signs), with HSP-Trees having an accuracy of 72% against 23.56% for the SP-Trees and 73.7% for HMMs. However, the computational simplicity of the HSP-Trees resulted in a sequence of 981 signs requiring only 2 minutes for processing against 20 minutes for HMMs. Analysis of the output results for both Dataset I and II showed that the errors of HSP-Trees occured at frames around the boundaries between two signs.

The results for Database III can be seen in Fig. 5c. Here, we show results across different top-N signs (x-axis). As can be seen, the proposed method obtains an average accuracy of 71.4% with an average of 8 false positives when a winner-takes-all approach is employed. The results were obtained using only small numbers of training instances, co-articulation factors and importantly without the use of any language model.



(a) Subject Dependent Accuracy Results (b) Subject Independent Accuracy Results (c) Boston-104 results

Figure 5: Sign sequence accuracy for the Dataset I test sequences for both (a) subject dependent and (b) subject independent results. The true positive and false detections for different top-N signs of Dataset III is shown in (c).

7. Conclusions and Future Work

This article presented a new framework for learning the temporal signatures that characterise signs, applied to the task of spotting signs from a defined lexicon in an unsegmented video. The proposed method is efficient, robust to unseen users and allows for sharing discrimative subpatterns between learnt signs. This framework extends on the state-of-the-art SP-Trees approach [8] as follows: 1) an extension of SPs called SIPs, that includes interval information, increasing the discriminative power of learnt patterns. 2) a tree structure called HSP-Trees allowing generic subsequence sharing between patterns (SP-Trees only allow sharing between patterns with the same initial sequence). 3) an efficient algorithm for learning HSP-Trees using subsequence sharing between classes for learning on large numbers of examples and categories (981 classes). Evaluation of the method on continuous sign sentences, demonstrates that it can cope with co-articulation. The proposed approach was shown experimentally to yield significantly higher performance than SP-Trees for unsegmented SLR, in both signer dependent (49 % improvement) and independent datasets (12% improvement). Additionally, comparisons with HMMs have shown that the proposed method either equal or exceed the accuracy of HMMs, in signer dependent (71% HSP vs 63% HMM) and signer independent (54%HSP vs 49%HMMs), with significantly reduced processing time: (2minutes HSP vs 20 minutes HMMs for processing 981 signs). Future work will concentrate on eliminating the spurious labels at sign boundaries, integration of language models into the HSP-Trees and extensive evaluations on other continuous sign datasets.

Acknowledgements

This work was funded by the UK government.

References

[1] P. Dreuw, D. Rybach, T. Deselaers, M. Zahedi, and H. Ney. Speech recognition techniques for a sign language recognition system. In *Proc. of Interspeech*, pages 2513–2516, 2007.

[2] R. Elliott, H. Cooper, J. Glauert, R. Bowden, and F. Lefebvre-Albaret. Search-by-example in multilingual sign language databases. In *2nd Intl. Workshop on Sign Language Translation and Avatar Technology (SLTAT)*, Oct. 23 2011.

[3] M. Elmezain. A hidden markov model-based continuous gesture recognition system for hand motion trajectory. In *ICPR*. IEEE, 2008.

[4] Z. Gavrilov, S. Sclaroff, C. Neidle, and S. Dickinson. Detecting reduplication in videos of american sign language. In *Proc. Int. Conf. on Language Resources and Evaluation (LREC)*, 2012.

[5] T. Kadir, R. Bowden, E. Ong, and A. Zisserman. Minimal training, large lexicon, unconstrained sign language recognition. volume 2, pages 939 – 948.

[6] R.-H. Liang and M. Ouhyoung. A real-time continuous gesture recognition system for sign language. In *Face and Gestures*. IEEE, 1998.

[7] L.-P. Morency, A. Quattoni, and T. Darell. Latent-dynamic discriminative models for continuous gesture recognition. In *CVPR*. IEEE, 2007.

[8] E. Ong, H. Cooper, N. Pugeault, and R. Bowden. Sign language recognition using sequential pattern trees. In *CVPR*. IEEE, 2012.

[9] V. Pitsikalis, S. Theodorakis, C. Vogler, and P. Maragos. Advances in phonetics-based sub-unit modeling for transcription alignment and sign language recognition. In *Gesture Recognition*.

[10] D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, and H. Ney. The RWTH aachen university open source speech recognition system. In *10th Annual Conference of the International Speech Communication Association*, pages 2111–2114, 2009.

[11] H. Wang, A. Stefan, S. Moradi, V. Athitsos, C. Neidle, and F. Kamangar. A system for large vocabulary sign search. In *ECCV International Workshop on Sign, Gesture, and Activity*, Crete, Greece, Sept. 2010.

[12] H.-D. Yang, S. Sclaroff, and S.-W. Lee. Sign language spotting with a threshold model based on conditional random fields. 31(7):1264 – 1277, July 2009.