Sign Language Recognition using Sequential Pattern Trees

Eng-Jon Ong Helen Cooper Nicolas Pugeault Richard Bowden
The Centre for Vision, Speech and Signal Processing,
University of Surrey, Guildford GU2 7XH, Surrey, UK
e.ong,h.cooper,n.pugeault,r.bowden@surrey.ac.uk

Abstract

This paper presents a novel, discriminative, multi-class classifier based on Sequential Pattern Trees. It is efficient to learn, compared to other Sequential Pattern methods, and scalable for use with large classifier banks. For these reasons it is well suited to Sign Language Recognition. Using deterministic robust features based on hand trajectories, sign level classifiers are built from sub-units. Results are presented both on a large lexicon single signer data set and a multi-signer Kinect™data set. In both cases it is shown to out perform the non-discriminative Markov model approach and be equivalent to previous, more costly, Sequential Pattern (SP) techniques.

1. Introduction

This paper attempts to tackle the problem of independent sign-language recognition. Sign Language, being as complex as any spoken language, has many thousands of signs each differing from the next by minor changes in hand motion, shape or position. Its grammar includes the modification of signs to indicate an adverb modifying a verb and the concept of placement where objects or people are given a spatial position and then referred to later. This, coupled with the intra-signer differences make true Sign Language Recognition (SLR) an intricate challenge. Previous SLR work has shown the advantage of using tracking-based, sub-unit classifiers [6]. While others have shown results on larger datasets using data driven approaches. Wang *et al*., created an American Sign Language (ASL) dictionary based on similarity between signs using a Dynamic Space-Time Warping (DSTW) approach. They used an exemplar, sign level approach and did not use Hidden Markov Models (HMMs) due to the high quantities of training data required. They present results for a dictionary containing 1113 signs [12]. More recently, 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 over 961 signs. The common requirement of tracking means that the Kinect™offers the sign recognition community a short-cut to real-time performance. Zafrulla *et al*. used this to extend their previous CopyCat game for deaf children [13]. By using a cross validated system they trained HMMs to recognise signs. One disadvantage of HMMs is that they are learnt in a non-discriminatory fashion. As a result, during the learning process, data from alternate classes are ignored. This can result in sub-optimal classifiers, particularly when there are large ambiguities between different classes. Additionally, HMMs do not perform explicit feature selection. As a result, features that do not contribute or are detrimental to the recognition process are always included.

To address the above issues, we consider discriminative spatio-temporal patterns for classification called sequential patterns (SPs). SPs are sequences of feature subsets. Using SPs provides the advantage of explicit spatio-temporal feature selection. Additionally, SPs do not require dynamic time warping for temporal alignment. A set of SPs can also be stored and later used efficiently within a tree structure. Research in SPs mainly lie in the area of data mining, where the objective is finding frequent SPs. Related to our work is Ayres et al. [1], where SPs are organised into a prefix tree, which is then used to mine frequent patterns; the tree itself is discarded upon completion of the mining process. Later, Hong et al. [5] proposed a method for mining frequent patterns by means of incrementally updating to a tree structure based on shared subsequences between frequent SPs. It should be noted that [1, 5] do not consider the problem of classification; simply mining of SPs that frequently occur over a set of unlabelled examples. The use of SPs for classification of signs was recently proposed by Elliott et al[4], where SPs were learnt in a discriminatory fashion before being combined into strong classifiers for recognising signs. One major drawback of using SPs for classification is that they only permit binary classifiers to be built. For problems with more than 2 classes, it is necessary to employ 1vs1 classifiers within a voting framework; a method that does not scale well with the number of classes.

1.1. Contributions

The work presented in this paper has the following key contributions. Firstly, we improve on existing SP based approaches in the following ways: We introduce a novel Sequential Pattern Tree that is multi-class in nature. Allowing us to tackle the binary limitation of existing SP classifiers. Importantly, the tree structured model facilitates spatio-temporal feature sharing across different classes. This, along with the multi-class aspect of the Sequential Pattern Tree (SP-Tree), ultimately results in a classifiers that are considerably simpler but with superior or state-of-the-art recognition performance. Additionally, the classifiers now scale very well with the number of classes. We show that the run-time efficiency of our classifiers is *independent* of the number of classes. Instead, it only depends on the depth and number of SP-Trees used.

1.2. Overview

The rest of the paper is organised as follows: Section 2 introduces the novel Sequential Pattern Trees. Here, we will also provide an important theoretical link between SP-Trees and SPs. Following this, Section 3 proposes a novel algorithm for efficiently learning a set of SP-Trees within the Boosting framework. Following this, we apply the proposed method in experiments investigating both signer dependent and independent datasets in Section 4. We also investigate the scalability of our approach by testing on a dataset with a large number of signs (982). We then compare our results with a traditional Markovian approach as well as state-of-the-art SP Boosting and HMMs methods before concluding in Section 5

2. Sequential Pattern Trees

In this section, we propose a novel model known as *Sequential Pattern Tree* for efficiently representing discriminative spatio-temporal patterns that can be used later for performing multi-class recognition.

2.1. Sequential Patterns

The concept of an SP-Tree is based on SPs. A SP is a sequence of feature subsets, where these features are required to take a particular value. Here, since only binary feature vectors are considered, this required feature value is set as 1. Thus, a SP can be specifically defined as:

Definition 2.1 *Given a binary feature vector $F = (f_i)_{i=1}^{D}$, let $T \subset \{1, ..., D\}$ be a set of integers where $\forall t \in T, f_t = 1$, that is, T represents all the dimensions of F that have the value of 1. We call T an* itemset*. Let $\mathbf{T} = (T_i)_{i=1}^{|\mathbf{T}|}$ be a sequence of $|\mathbf{T}|$ itemsets. We denote $\mathbf{T}$ as a* SP*. (Examples shown in Figure 1).*

In order to use SPs for classification, we firstly note that an input sequence of T consecutive D-dimensional binary vectors can be converted into a T length SP. Next, we define an operator to determine if a SP is present within a given feature vector sequence:

Definition 2.2 *Let $\mathbf{T}$ and $\mathbf{I}$ be SPs. We say that the SP $\mathbf{T}$ is present in $\mathbf{I}$ if there exists a sequence $(\beta_i)_{i=1}^{|\mathbf{T}|}$, where $\beta_i < \beta_j$ when $i < j$ and $\forall i = \{1, ..., |\mathbf{T}|\}, T_i \subset I_{\beta_i}$. This relationship is denoted with the $\subset_S$ operator, i.e. $\mathbf{T} \subset_S \mathbf{I}$. Conversely, if the sequence $(\beta_i)_{i=1}^{|\mathbf{T}|}$ does not exist, we denote it as $\mathbf{T} \not\subset_S \mathbf{I}$.*

2.2. SP-Tree Definition

In this section, we provide a mathematical definition of an SP-Tree as well as how paths within this tree result in different SPs. To start, we propose to define each node in the SP-Tree such that it can assume two potential roles: a feature selector, or if the node is a leaf node, a output a class label.

Definition 2.3 *Let $c \in \{1, ...C\}$ be a class label, $d \in \{1, ..., D\}$ be a feature vector dimension index. We define a* node *N of an SP-Tree as the following pair: $N = (c, d)$. In the future, we will refer to c as the* node label *and d as the* node dimension*. We denote the space of nodes with $\mathcal{N}$.*

Next, we attempt to capture both the spatial and temporal aspects of the data by proposing two different type of edges for connecting a pair of SP-Tree nodes: static edges and sequential edges.

Definition 2.4 *Let $N_1 = (c_1, d_1)$,$N_2 = (c_2, d_2)$, $N_1, N_2 \in \mathcal{N}$ be two nodes in an SP-Tree. We define an* edge *E of an SP-Tree as the tuple: (N_1, N_2, e, k), where $e \in 1, 2$ denotes whether E is a* static edge *($e = 1$) or a* sequential edge *($e = 2$) and $k \in +1, -1$, where $k = 1$ denotes a positive (i.e. true) decision edge, and $k = -1$ represents a negative (i.e. false) decision edge. In the case where E is a positive static edge, then the itemset $\{c_1, c_2\}$ is captured by the respective SP-Tree, or $\{c_2\}$ when E is a negative static edge. When E is a positive sequential edge, the SP $(\{c_1\}, \{c_2\})$ is captured by the SP-Tree. Conversely, with a negative sequential edge, the SP $(\{c_2\})$ is captured instead. We denote the space of SP-Tree edges as $\mathcal{E}$.*

Having defined both the tree nodes and their edges, we now provide the definition for a SP-Tree:

Definition 2.5 *Let $\mathbf{N} = \{N_i : N_i \in \mathcal{N}\}_{i=1}^{|\mathbf{N}|}$ be a set of SP-Tree nodes and $\mathbf{E} = \{E_i : E_i \in \mathcal{E}\}_{i=1}^{|\mathbf{E}|}$ be a set of SP-Tree edges. We define the* root node *as $N' \in \mathbf{N}$ such that $(N, N') \notin \mathbf{E}$ for each $N \in \mathbf{N}$. A* SP-Tree *$\mathbf{T}$ is defined as: $\mathbf{T} = (\mathbf{N}, \mathbf{E}, N')$. We also define* leaf nodes *of T as any node N where there does not exist any $M \in \mathbf{N}$ such that $(N, M) \in \mathbf{E}$.*

In this work, we only consider binary SP-Trees. This imposes constraints on the possible edges that can exist between pairs of nodes, specifically: given a SP-Tree $\mathbf{T} = (\mathbf{N}, \mathbf{E})$, for each node $N \in \mathbf{N}$ that is not a leaf node, there are only 2 edges leading from N, namely $E^+, E^- \in \mathbf{E}$ where $E^+ = (N, M, e^+, +1)$ and $E^- = (N, L, e^-, -1)$ for some $e^+, e^- \in \{1, 2\}$ and $M \neq L$. Additionally, each node N has at most one parent node: $|\{(M, N) : (M, N) \in \mathbf{E}, M \in \mathbf{N}, M \neq N\}| \leq 1$.

2.3. Multi-Class (Weak) Classfication using SP-Trees

In this section, we describe how one can use an SP-Tree to perform multi-class classification of given input sequences. To achieve this, we first need to see how SPs are related to SP-Trees. To this end, we firstly consider the types of paths one can make when traversing between two nodes on the tree.

Definition 2.6 *We define a* path $\mathbf{P}$ *through an SP-Tree* $\mathbf{T}$ *as a sequence of nodes* $\mathbf{P} = (N_i)_{i=1}^{|P|}$ *where for each* $i \in \{1, ..., |P| - 1\}$*, there exists some* $e_i \in \{1, 2\}$ *and some* $k_i \in -1, +1$ *such that* $(N_i, N_{i+1}, e_i, k_i) \in \mathbf{E}$*. We define a specific path called a* static path *if for each* $i \in \{1, ..., |P|\}, e_i = 1$.

Given the above definition, we now show the relationship between static paths and itemsets.

Lemma 2.1 *Let* $\mathbf{P} = (N_i)_{i=1}^{|P|}$ *be a static path from the SP-Tree* $\mathbf{T}$ *and let* d_i *be the node dimension of node* N_i*. Let subset* $\mathbf{P}' \subseteq \mathbf{P}$ *be the set of nodes in* $\mathbf{P}$ *that are parents to positive edges linking two nodes in* $\mathbf{P}$*:* $P' = \{P : P \in \mathbf{P}, Q \in \mathbf{P}, (P, Q, 1, +1) \in \mathbf{E}\}$*. Then, we find that* $\mathbf{P}$ *models the itemset:* $I = \{d'_i\}_{i=1}^{|P'|}$*, where* d'_i *are the node dimensions of the nodes in* $\mathbf{P}'$*. Henceforth, we shall say that* I *is an itemset* derived *from static path* $\mathbf{P}$*. (The proof follows directly from the definition of static edges and Algo. 1.)*

Lemma 2.1 implies that if a path $\mathbf{P}$ contains C sequential links within it, then there are $C + 1$ static paths in P. From this, we now arrive at the important property that a unique SP can be derived from an SP-Tree path:

Theorem 2.2 *A SP of length* $B + 1$ *can be derived from an SP-Tree (*$\mathbf{T}$*) path* $\mathbf{P} = (N_i)_{i=1}^{|P|}$ *with* $B \geq 0$ *positive sequential links.*

Proof Since $\mathbf{P}$ has B positive sequential links, $\mathbf{P}$ is partitioned into a sequence of $B+1$ static paths: $(P_i)_{i=1}^{B+1}, P_i \subset P$. Let I_i be the itemset derived from static path P_i. Then, the itemset sequence $(I_i)_{i=1}^{B+1}$ is a SP.

An illustration of how different SP-Tree paths can produce different SPs can be seen in Figure 1.

2.3.1 Classification Algorithm

It is now possible to describe how SP-Trees can be used to classify a new input sequence within a multi-class problem. We start by proposing the function SPT_Path in Algo. 1 for extracting an SP-Tree path from an existing SP-Tree given an input sequence. Algo. 1 works in a greedy manner, whereby SPs corresponding to paths along the positive edges of the tree are given preference in the searching process. In other words, Algo. 1 will attempt to find SPs on paths with as many positive edges as possible, only using negative paths when there is no other choice. One crucial aspect of this algorithm is that only a *single* path in an SP-Tree is extracted for a given input sequence. Another important aspect is that the complexity of this algorithm in extracting the respective SP is O(n) with respect to the depth of the tree. In Section 3.4, we shall see how this is key in making our proposed method highly efficient, when compared against 1vs1 voting classifiers in a multi-class problem.

Next, given an input sequence $\mathbf{F}$ and an SP-Tree $\mathbf{T}$, let $\mathbf{P} = SPT\_Path(\mathbf{F}, \mathbf{T})$, where $\mathbf{P} = (P_i)_{i=1}^{|\mathbf{P}|}$. We can then obtain a class label for an SP-Tree path using the label of the last node in the path:

$$L(\mathbf{P}) = c^{|\mathbf{P}|} \tag{1}$$

where $P_{|\mathbf{P}|} = (c^{|\mathbf{P}|}, d^{|\mathbf{P}|})$, and $c \in \{1, ..., C\}$, where C is the number of classes. Using Algo. 1 and Eq. 1, let $\mathcal{F}$ be the space of input sequences, it is now possible to classify an input sequence ($\mathbf{F} \in \mathcal{F}$), given an SP-Tree $\mathbf{T}$ using the classification function $h^{\mathbf{T}} : \mathcal{F} \rightarrow \{1, ..., C\}$:

$$h^{\mathbf{T}}(\mathbf{F}) = L(SPT\_Path(\mathbf{F}, \mathbf{T})) \tag{2}$$

3. Boosting Discriminative SP-Trees

Having defined the SP-Trees and how one uses them for multi-class classification, we now deal with the problem of learning and combining the required trees to produce a robust and accurate classifier that can work with novel input sequences. To this end, we propose a novel machine learning method for learning strong classifiers based on SP-Trees within the Multi-class AdaBoost framework [14]. A strong classifier outputs a class label based on the maximum votes cast by a number (S) of selected and weighted weak classifiers: $H(I) = \arg\max_c \sum_{i=1}^{S} \alpha_i \mathbb{I}(h_i(I) = c)$. In this paper, the weak classifiers h_i are the SP-Tree classifiers defined in Section 2.3 as Eq. 2. Each h_i is selected iteratively with respect to the following error:

$$\epsilon_i = \sum_{i=1}^{X} \mathbb{I}(h_i(X_i) \neq y_i) \tag{3}$$

Typically, in order to determine the optimal weak classifier at each Boosting iteration, the common approach is to

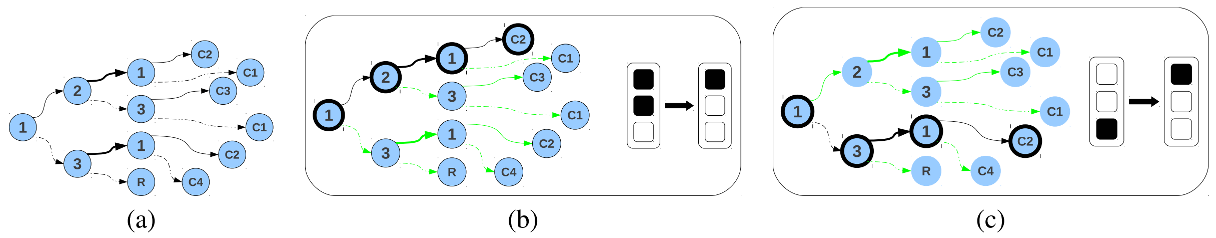


Figure 1: (a) An example of an SP-Tree with static (thin) and sequential (thick) edges. Each node has a +ve (solid line) and -ve (dashed) edge. Leaf nodes output class labels. (b) A tree path and its corresponding SP $(\{1,2\},\{1\})$ and belonging to class 2. (c) Another path belonging to class 2 on the same tree modelling SP $(\{3\},\{1\})$.

Algorithm 1 SP-Tree Path: $\mathbf{P}$ = *SPT_Path*($\mathbf{F}, \mathbf{T}$)

```
*Input*: Input sequence $\mathbf{F} = (F_i)_{i=1}^{F}, F_i \in \{0,1\}^D$
*Input*: SP-Tree $\mathbf{T} = (\mathbf{N}, \mathbf{E}, N')$, where the root node, set
of nodes and edges for $\mathbf{T}$ are $N'$, $\mathbf{N}$ and $E$ respectively.
*Output*: SP-Tree Path $\mathbf{P} = (P_i)_{i=1}^{|\mathbf{P}|}$
Initialise index set $R = \{1, ..., F\}$
Initialise current node to root node: $N^{cur} = N'$
$e^{cur} = 1$
$\mathbf{P} = (N'), A = 2$
**while** $N^{cur}$ is not leaf node **do**
  $d = d^{cur}$, where $d^{cur}$ is the node dimension of $N^{cur}$
  $G = \{j : j \in R, F_{j,d} = 1\}$
  **if** $G = \emptyset$ **then**
    $N^{cur} = M, e^{cur} = e$, such that $(N, M, e, -1) \in \mathbf{E}$
  **else**
    $N^{cur} = M, e^{cur} = e$, such that $(N, M, e, +1) \in \mathbf{E}$
    **if** $e^{cur} = 2$ **then**
      $R_{new} = \{(min(R) + 1), ..., F\}$
    **else**
      $R_{new} = G$
    **end if**
    $R = R_{new}$
  **end if**
  $P_A = N^{cur}, A = A + 1$
**end while**
Return $\mathbf{P} = (P_i)_{i=1}^{A}$
```

exhaustively consider the entire set of possible weak classifiers and finally select the best weak classifier (i.e. that with the lowest ϵ_i). However, when dealing with SP-Tree-based weak classifiers, the weak classifier search space becomes too large, making an exhaustive search no longer feasible. To address this, we next propose a novel method (detailed in Algo 2 and Algo 3) for efficiently learning a locally optimal SP-Tree-based weak classifier that also accounts for the training examples weight distribution.

3.1. Training Data Definitions

We firstly provide some preliminary definitions that we will re-use in the next few sections. Firstly, the number of classes is denoted as C, the training set with X number of training examples is defined as: $\mathbf{X} = \{X_i\}_{i=1}^{X}$, where X_i is a sequence of D-dimensional binary feature vectors. The length of X_i is denoted as $|X_i|$. We define $X_i = (x_i)_{i=1}^{|X_i|}$, $x_i \in \{0,1\}^D$. Associated with the examples in $\mathbf{X}$ is a set of labels $Y = (y_i)_{i=1}^{X}$, $y_i \in \{1, ..., C\}$ and a normalised set of example weights $W = (w_i)_{i=1}^{X}$, where $\sum_{i=1}^{X} w_i = 1$.

3.2. Simultaneous Learning of SP-Tree Structure and Features

Learning an SP-Tree weak classifier requires us to address the following issues: determining the label of the node; quantifying the "goodness" of a node; determining the node dimension; determining the edge type (i.e. static or sequential) between two nodes.

Firstly, we determine the label of a node as follows: Suppose we are considering an SP-Tree's ($\mathbf{T}$) node $N = (c, d)$ and are given a training subset ($\mathbf{X}' \subseteq \mathbf{X}$) called the *node training set*. Let the labels and weights of the examples in $\mathbf{X}'$ be Y' and W' respectively. A normalised histogram $(f_i)_{i=1}^{C}$ built using Y' will give us the label distribution of $\mathbf{X}'$. The label for N is then simply the maximum frequency label in $(f_i)_{i=1}^{C}$: $c = \arg\max_{i \in \{1,...,C\}} f_i$.

Determining the node dimensions and edge types in an SP-Tree is done simultaneously and in a top-down recursive manner, where we extend branches from leaf-nodes by giving each leaf node a valid node dimension and subsequent child leaf nodes. To achieve this, consider a leaf node N. When using Algo. 1, it can be seen that only a subset of training examples will "reach" it. Specifically, this training example subset is defined as $\mathbf{X}' = \{X : X \in \mathbf{X}, \mathbf{P} = SPT\_Path(X, \mathbf{T})\}$, where $\mathbf{X}' \subseteq \mathbf{X}$ and $\mathbf{P}$ be the path from the SP-Tree root node to N. Next, we observe that giving different values of d to node N and setting different edge types to the link between N and its parent will cause different binary partitions of $\mathbf{X}'$.

To see this, we find that $\mathbf{P}$ corresponds to some SP: $\mathbf{T} = (T_i)_{i=1}^{|T|}$ (Theorem 2.2), where T_i is the i^{th} itemset of $\mathbf{T}$. Suppose we want to set the node dimension of N as d and set the edge between N and its parent node M as a static edge. Then, the subset $\mathbf{X}'$ will be split into $\mathbf{X'}_+ = \{X : X \in \mathbf{X}', (T_1, ..., T_{|T|} \cup \{d\}) \subset_S X\}$ and $\mathbf{X'}_- = \{X : X \in \mathbf{X}', (T_1, ..., T_{|T|} \cup \{d\}) \not\subset_S X\}$, $T_{|T|} \cup \{d\}$ denotes the new itemset whereby d was added into itemset $T_{|T|}$.

When the edge between N and its parent node is a sequential edge, we have $\mathbf{X'}_+ = \{X : X \in \mathbf{X}', (T_1, ..., T_{|T|}, \{d\}) \subset_S X\}$ and $\mathbf{X'}_- = \{X : X \in \mathbf{X}', (T_1, ..., T_{|T|}, \{d\}) \not\subset_S X\}$. Here, the new SP $(T_1, ..., T_{|T|}, \{d\})$ is simply $\mathbf{T}$ with a new itemset $\{d\}$ appended at the end. Note, that in both edge cases, $\mathbf{X}' = \mathbf{X'}_+ \cup \mathbf{X'}_-$.

It is now possible to measure how "good" a node split is. To this end, we utilise an adapted Gini impurity criteria commonly used in decision tree learning [2]. Here, we have changed the criteria to account for the boosted example weights. Suppose we have found that node N has caused a partition $\mathbf{X'}_+$ and $\mathbf{X'}_-$. Suppose the corresponding weights of these partitions are W'_+ and W'_- respectively. Similarly, let the corresponding labels be Y'_+ and Y'_- respectively. We also define the total positive and negative partition weight coefficient as: $Z^+ = \sum_{i=1}^{|W'_+|} w'_{+,i} / \sum_{i=1}^{|W'|} w'_i$ and $Z^- = \sum_{i=1}^{|W'_-|} w'_{-,i} / \sum_{i=1}^{|W'|} w'_i$ respectively. Additionally, using both the weights and labels, we can compute a normalised label histogram for each partition: F'_+ and F'_- respectively. The split criteria is defined as:

$$\gamma = Z^+(1 - \sum_{i=1}^{C} f_{+,i}^2) + Z^-(1 - \sum_{i=1}^{C} f_{-,i}^2) \qquad (4)$$

It is now possible to combine all the above methods into a SP-Tree learning algorithm given in Algo. 2. This algorithm learns an SP-Tree in a greedy manner, whereby the splitting criteria defined in Eq. 4 is recursively minimised at each non-leaf tree-node. We start by determining the optimal node dimension for the root node with respect to Eq. 4. This induces a partitioning of the dataset into two training subsets. Each training subset is then passed on to the a new child node. From this point onwards, the algorithm attempts obtain the optimal node dimension and edge type for the respective child node, again with respect to Eq. 4. The optimal node dimension and edge type is then used to configure the child node being considered. This process is repeated in a recursive manner via a queue-based system until one of 3 termination criteria is reached: 1) maximum tree-depth β is reached; 2) training subset is smaller than minimum size α (set here as 1); 3) The training subset is "pure" (i.e. only belongs to a single class).

Algorithm 2 SP-Tree Learn Algorithm

```
Input: Training Set: $\{(\mathbf{X}_i, y_i, w_i)\}_{i=1}^{X}$ (Section 3.1)
Output: SP-Tree $\mathbf{T}$
Queue element: $(Node, TrainSubset, SearchDims., Depth)$
Set root node dim to $d \in \{1, ..., D\}$ s.t. $\mathbf{X}$ gives optimal
$\gamma$ (Eq. 4).
Let root node $N'$ partition $\mathbf{X}$ into $\mathbf{X}_+$ and $\mathbf{X}_-$.
Get root label: $c$ using $Y$ (Section 3.2).
Root node: $N' = (c, d)$.
$N'$ leaf nodes: $L = (-1, -1)$ and $M = (-1, -1)$.
$\mathbf{N} = \{N', L, M\}$
$\mathbf{E} = \{(N', L, -1, -1), (N', M, -1, +1)\}$.
Initial search dimensions: D' = {1,...,D} - {d}
$Q = \{(L, \mathbf{X}_+, D', 2), (L, \mathbf{X}_-, D', 2)\}$.
while $Q \neq \emptyset$ do
    Remove last item of $Q$: $(N^{cur}, \mathbf{X}^{cur}, D^{cur}, O^{cur})$
    if $|\mathbf{X}^{cur}| \leq \alpha$ OR $depth^{cur} \geq \beta$ then
        break
    end if
    Denote the edge from $N^{cur}$ to its parent node $(M^{cur})$
    as $E^{cur}$.
    1) Get optimal static edge node dimension:
        Set $E^{cur} = (M^{cur}, N^{cur}, 1, k)$
        Get $d^{stat} \in D^{cur}$ s.t. $\mathbf{X}^{cur}$ gives min $\gamma^{stat}$ (Eq. 4).
        Partitions of $d^{stat}$ and $E^{cur}$: $\mathbf{X}_+^{stat}$ and $\mathbf{X}_-^{stat}$.
    2) Get optimal sequential edge node dimension:
        Set $E^{cur} = (M^{cur}, N^{cur}, 2, k)$
        Set $d^{seq} \in \{1, ..., D\}$ s.t. $\mathbf{X}^{cur}$ gives min $\gamma^{seq}$ (Eq.
    4).
        Partitions of $d^{seq}$ and $E^{cur}$: $\mathbf{X}_+^{seq}$ and $\mathbf{X}_-^{seq}$.
    Set label of $N^{cur}$ as $c^{cur}$
    if $\gamma^{stat} \leq \gamma^{seq}$ then
        Update current node: $N^{cur} = (c^{cur}, d^{stat})$
        Update $E^{cur} = (M^{cur}, N^{cur}, 1, k)$
        New search dims: $D^{new} = D^{cur} - \{d^{stat}\}$
        New partitions: $\mathbf{X}_-^{cur} = \mathbf{X}_-^{stat}$ and $\mathbf{X}_+^{cur} = \mathbf{X}_+^{stat}$.
    else
        Update current node: $N^{cur} = (c^{cur}, d^{seq})$
        Update $E^{cur} = (M^{cur}, N^{cur}, 2, k)$
        New search dims: $D^{new} = \{1, ..., D\}$
        New partitions: $\mathbf{X}_-^{cur} = \mathbf{X}_-^{seq}$ and $\mathbf{X}_+^{cur} = \mathbf{X}_+^{seq}$.
    end if
    if $min(\gamma^{stat}, \gamma^{seq}) > 0$ then
        New nodes: $L = (-1, -1)$ and $K = (-1, -1)$
        $\mathbf{N} = \mathbf{N} \cup \{L, K\}$.
        $\mathbf{E} = \{(N^{cur}, L, -1, -1), (N^{cur}, K, -1, +1)\} \cup \mathbf{E}$.
        New depth: $O^{new} = O^{cur} + 1$
        $Q = Q \cup \{(K, \mathbf{X}_+^{cur}, D', O^{new}), (L, \mathbf{X}_-^{cur}, D', O^{new})\}$.
    end if
end while
Return SP-Tree: $\mathbf{T} = (\mathbf{N}, \mathbf{E})$
```

3.3. Final Algorithm

The final SP-Tree Boosting algorithm is detailed in Algo. 3. We have chosen to iteratively learn new SP-Trees based on the multi-class AdaBoost method. However, we are not limited to this particular form of Boosting and it would be easy to integrate the SP-Tree learning algorithm (Algo. 2) into other Boosting methods (e.g. GentleBoost, etc...).

Algorithm 3 SP-Tree-Boost Algorithm

```
Initialise example weights: $\forall w_i \in W, w_i = 1/X$
**for** $t = 1, ..., M$ **do**
    Select ($h_t = h^{\mathbf{T}_{best}}$) using Algo. 2
    Obtain the classification error $\epsilon_t$ for $h_t$ (Eq. 3)
    Obtain the weight $\alpha_t = \ln \frac{1-\epsilon_{best}}{\epsilon_{best}} + \ln(C-1)$
    Update weights: $w_i = w_i \exp(-\alpha_i [h_t(X_i) \neq y_i])$
    Normalise weights: $\sum_{i=1}^{X} w_i = 1$
**end for**
Return the strong classifier $H(X) = \arg\max_c \sum_{i=1}^{M} \alpha_i \mathbb{I}(h_i(X) = c)$
```

3.4. SP-Tree Classifier Run-time Complexity

One major advantage of using strong classifiers based on SP-Trees for multi-class recognition is in the run-time complexity. Firstly, we notice that the run-time complexity of the weak classifiers is directly dependent on the complexity of the Algo. 1 which has a run-time complexity equal to depth of the tree. Crucially, this is *independent* of the number of classes. As an example, suppose the SP-Trees were limited to depth 15. An evaluation of any input sequence for a weak classifier will at most traverse 15 nodes in its associated SP-Tree. Thus, the complexity of the final strong classifier of M classifiers with SP-Trees limited to depth K is simply: $O(MK)$. In contrast, the 1vs1 voting-based classifiers used for converting binary classifiers into multi-class recognisers has complexity of $O(MC^2)$. As a result, our proposed classifiers permit massive efficiency gains over traditional 1vs1 classifiers, especially when the number of classes grow large (e.g. 100s).

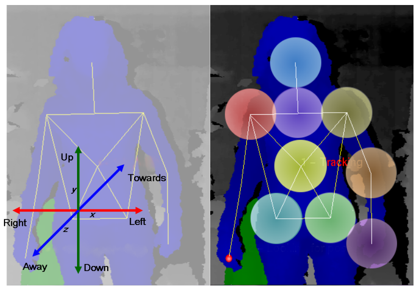
4. Experiments

The experiments are designed to show the benefits of using a discriminative learning model which performs feature selection. This should enable it to be more robust to noise in complex data. Thus, we use one data set with multiple signers in real world scenarios. However, while also being robust, the approach needs to be scalable, therefore, the SP-Tree is also demonstrated on a large lexicon set.

4.1. Database I: DGS Kinect 40

The first dataset was captured using a Kinect™camera. It contains 40 Deutsche Gebärdensprache - German Sign Language (DGS) signs; a random mix of both similar and dissimilar signs. There are 14 participants each performing all the signs 5 times. The data was captured using a mobile system. All signers in this set are non-native giving a wide variety of signing styles.This set-up results in a complex data set with examples of both inter and intra signer differences.

Motion features extracted from this data are inspired by sign linguistics conceptual terms; *e.g.* 'hands move left' 'dominant hand moves up' [11]. Using a tracked human skeleton from the OpenNI framework [8] 3D trajectories of the hands can be extracted. The types of motion are derived directly from deterministic rules on the x,y and z co-ordinates of the hand positions. These are then described using binary features based on single and dual handed motions (see Figure 2a for example motions). Linguistic location characteristics are also abstract from the base x and y co-ordinates; they happen in relation to the signer such as 'at the head' or 'by the elbow'. As such, location features are calculated using the Euclidean distance of the dominant hand from skeletal joints as shown in Figure 2b. When the dominant hand moves into the joint region then that feature is activated. The final feature vector representing a single frame is a 24-dimensional binary vector.

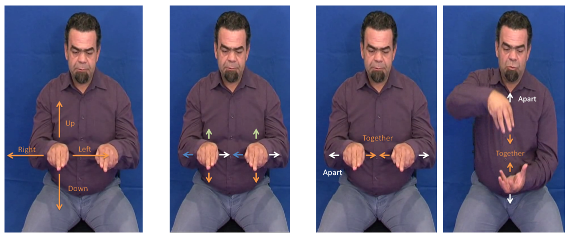


(a) Directions (b) Locations

Figure 2: Kinect Features

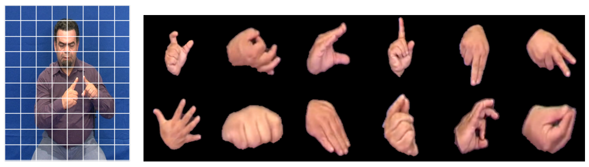
4.2. Database II: GSL 982 Signs

The second dataset contains videos of 982 Greek Sign Language (GSL) signs with 5 examples of each, performed by a single native signer (giving a total of 4910 samples). This large lexicon gives a wide range of signs, some contain similar motions and occasionally some signs share component parts. Hand and head trajectories are extracted from the videos using the work of Roussos *et al.* [10]. Three types of features are extracted: motion of the hands; location of the sign being performed; and handshape used. As with the pre-



(a) Single handed (b) Bimanual: Synchronous (c) Bimanual: Together/Apart

Figure 3: Motions detected from 2D tracking



(a) Grid Locations (b) Cannonical Handshapes

Figure 4: Locations and Handshapes from 2D Data

vious data, individual hand motions in the x plane (left and right) and y plane (up and down) are used. These are shown in Figure 3. For the location information, the x,y coordinates are quantised into a codebook based on the signer's head position and scale in the image. The quantised values fall in the range of $y' \in \{0..10\}$ and $x' \in \{0..8\}$ (for a standard signing space) as shown in Figure 4a. Finally handshapes are considered, the signers dominant hand is segmented and HOGs extracted. Decision forests are learnt over the HOG feature vector to classify the handshapes into the 12 different classes based on canonical linguistic forms, see Figure 4b [3]. These features are again represented as binary values and result in a 65 dimension feature vector.

4.3. Experimental Setup

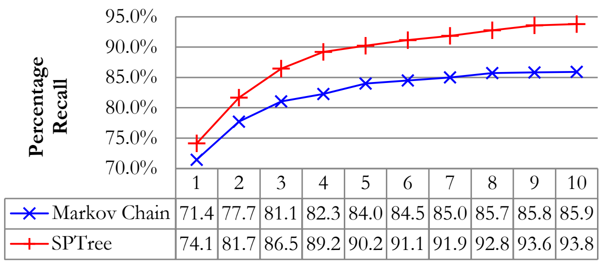
The accuracy results are presented as how often the correct result appears within the top n returned signs. For a given query q the results are ranked and the correct result appears at rank $R(q)$. However, the signer will only look through a few (n) signs before assuming that the result is not found. As such we define a measure of success $s(q, n)$ which is true when $R(q) \leq n$ and false otherwise. The percentage recall for a given value of n is therefore the proportion of queries where $s(q, n)$ is true. As with any search system, the higher up the results the correct sign appears, the better the ranking system. Results are shown for various values of n. Ideally accuracy should be high for low values of n allowing a user to find the sign quickly.

4.4. Results I: (Kinect 40)

As multiple signers are available in the DGS Kinect Dataset, signer independent experiments are performed. This involves reserving data from a single subject for testing, then training on the remaining signers. This process is repeated across all signers in the data set. For this experiment, 200 SP-Trees were used with max depth 30. The results of this are shown in Figure 5 for values of $n = 1$ and $n = 4$; the top 2.5% and 10% of the dataset. Compared are the results from processing the data with Markov Chains and SP Boosting [7]. Even given the relatively large number of signers in the data, both forms of SPs outperform the Markov Chains by about 5%. This is due to their discriminative nature which enables them to ignore the non-salient features without having to model them. The proposed approach achieves similar results to the SPBoosting, outperforming it by 1% when $n = 1$ and trailing by 1% when $n = 4$, but at a much greater efficiency. It will evaluate 200 SPs for each example where, in comparison, the SPBoosting uses $10(40^2 - 40) = 15600$ SPs for this 40 class problem. As such, it is possible to apply this approach to much larger datasets as shown in the next section.

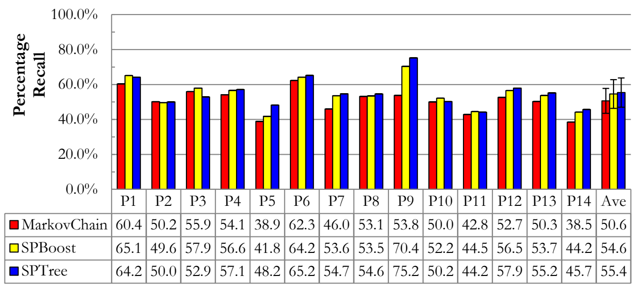
4.5. Results II: (982 sign database)

Since the SPBoosting is untenable on such a large dataset (requiring nearly a million SPs) the results shown here compare only to Markov Chains [3]. Here, 300 SP-Trees were used with max depth 30. As can be seen, even on this single signer data set (without the need to model inter-signer noise) the SP-Trees are able to outperform the Markov Chains. The trees are 2.7% more accurate when $n = 1$, becoming 7.9% better when $n = 10$. Work has been done on a subset of 961 signs from the same dataset by Pitsikalis *et al.* [9]. They use a more complex set of features based on linguistics and combine them using HMMs. In spite of this combination of strong elements, the SP-Trees are able to compete with the 62% they achieve for $n = 1$.



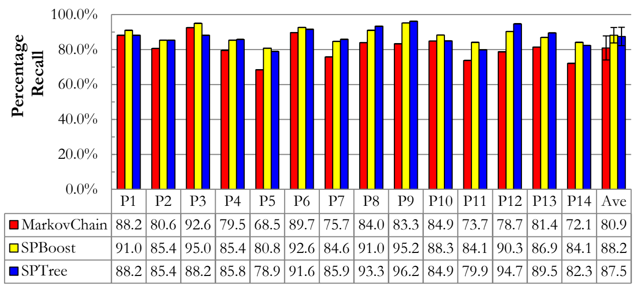
	1	2	3	4	5	6	7	8	9	10
Markov Chain	71.4	77.7	81.1	82.3	84.0	84.5	85.0	85.7	85.8	85.9
SPTree	74.1	81.7	86.5	89.2	90.2	91.1	91.9	92.8	93.6	93.8

Figure 6: Results on the 982 GSL sign dataset



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	Ave
🟥 MarkovChain	60.4	50.2	55.9	54.1	38.9	62.3	46.0	53.1	53.8	50.0	42.8	52.7	50.3	38.5	50.6
🟨 SPBoost	65.1	49.6	57.9	56.6	41.8	64.2	53.6	53.5	70.4	52.2	44.5	56.5	53.7	44.2	54.6
🟦 SPTree	64.2	50.0	52.9	57.1	48.2	65.2	54.7	54.6	75.2	50.2	44.2	57.9	55.2	45.7	55.4

(a) $n = 1$



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	Ave
🟥 MarkovChain	88.2	80.6	92.6	79.5	68.5	89.7	75.7	84.0	83.3	84.9	73.7	78.7	81.4	72.1	80.9
🟨 SPBoost	91.0	85.4	95.0	85.4	80.8	92.6	84.6	91.0	95.2	88.3	84.1	90.3	86.9	84.1	88.2
🟦 SPTree	88.2	85.4	88.2	85.8	78.9	91.6	85.9	93.3	96.2	84.9	79.9	94.7	89.5	82.3	87.5

(b) $n = 4$

Figure 5: Results on the 40 DGS sign Kinect dataset

5. Conclusions

This paper has presented a novel approach to using SPs for discriminative classification. By using a tree architecture the resulting classifiers are both faster to learn and more efficient to apply than previous SP approaches. This has been effectively demonstrated on two sign language data sets, showing both the robustness and scalability of the approach. By applying it to a 40 sign database of 14 signers the discriminative power of SPs is shown over Markov model implementation. The approach gains 55% as the first ranked sign and 87% within the top 10 signs. This is not only equivalent to other SP methods but is completed with significantly fewer individual SP comparisons making it faster and more efficient. Secondly the classifier is shown to be suitable for large classifier banks by applying it to a dictionary of 982 signs. For this task it is compared to only the Markov Model approach since other state of the art SP approaches are non-scalable. Here, the SP-Trees were able to outperform the Markov models resulting in a 7.9% increase when considering the top 1% of the ranked data set.

6. Acknowledgements

The research leading to these results has received funding from EU FP7 grant 231135 - DictaSign, EPSRC grant EP/I011811/1 Learning to Recognise Dynamic Visual Content and the UK Home Office.

References

[1] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick. Sequential pattern mining using bitmaps. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, July 2002.

[2] L. Brieman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression trees*. Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[3] H. Cooper, N. Pugeault, and R. Bowden. Reading the signs: A video based sign dictionary. In *Procs. of Wkshp : ARTEMIS workshopICCV*, Barcelona, Spain, Nov. 7 – 11 2011.

[4] R. Elliott, H. Cooper, J. Glauert, R. Bowden, and F. Lefebvre-Albaret. Search-by-example in multilingual sign language databases. In *Procs. of Second International Workshop on Sign Language Translation and Avatar Technology (SLTAT)*, Dundee, Scotland, Oct. 23 2011.

[5] T. Hong. Incrementally fast updated sequential pattern trees. In *Proceedings of International Conference on Machine Learning and Cybernetics*, pages 3991–3996, 2008.

[6] T. Kadir, R. Bowden, E. Ong, and A. Zisserman. Minimal training, large lexicon, unconstrained sign language recognition. In *Procs. of BMVC*, volume 2, pages 939 – 948, Kingston, UK, Sept. 7 – 9 2004.

[7] E.-J. Ong and R. Bowden. Learning sequential patterns for lipreading. In *Procs. of BMVC To Appear*, Dundee, UK, Aug. 29 – Sept. 10 2011.

[8] OpenNI organization. *OpenNI User Guide*, November 2010. Last viewed 20-04-2011 18:15.

[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 *Procs. of Int. Conf. CVPR Wkshp : Gesture Recognition*, Colorado Springs, CO, USA, June 21 – 23 2011.

[10] A. Roussos, S. Theodorakis, V. Pitsikalis, and P. Maragos. Hand tracking and affine shape-appearance handshape subunits in continuous sign language recognition. In *Procs. of Int. Conf. ECCV Wkshp : SGA*, Heraklion, Crete, Sept. 5 – 11 2010.

[11] R. Sutton-Spence and B. Woll. *The Linguistics of British Sign Language: An Introduction*. Cambridge University Press, 1999.

[12] 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.

[13] Z. Zafrulla, H. Brashear, P. Presti, H. Hamilton, and T. Starner. Copycat - center for accessible technology in sign. http://tinyurl.com/3tksn6s, Dec. 2010.

[14] J. Zhu, H. Zou, S. Rosset, and T. Hastie. Multi-class adaboost. *Statistics and Its Interface*, 2:349–360, 2009.