Scalable Action Recognition with a Subspace Forest

Stephen O'Hara and Bruce A. Draper
Colorado State University
1873 Campus Delivery
Fort Collins, CO 80523
{svohara, draper}@cs.colostate.edu

Abstract

We present a novel structure, called a Subspace Forest, designed to provide an efficient approximate nearest neighbor query of subspaces represented as points on Grassmann manifolds. We apply this structure to action recognition by representing actions as subspaces spanning a sequence of thumbnail image tiles extracted from a tracked entity. The Subspace Forest lifts the concept of randomized decision forests from classifying vectors to classifying subspaces, and employs a splitting method that respects the underlying manifold geometry. The Subspace Forest is an inherently parallel structure and is highly scalable due to $O(\log N)$ recognition time complexity. Our experimental results demonstrate state-of-the-art classification accuracies on the well-known KTH Actions and UCF Sports benchmarks, and a competitive score on Cambridge Gestures. In addition to being both highly accurate and scalable, the Subspace Forest is built without supervision and requires no extensive validation stage for model selection. Conceptually, the Subspace Forest could be used anywhere set-to-set feature matching is desired.

1. Introduction

A common approach to action recognition is the extraction of localized space-time features which are used in histogram matching [18] or in higher-level models involving feature covariances or hierarchies [2]. Local features-based approaches often require numerous design decisions and parameter selections (e.g., interest point selection, feature representation, quantization method, codebook size), and use heavy machine learning algorithms, such as Multiple Kernel Learning, to achieve good results. Concerns with complex approaches include overfitting and scaling to real-world problem domains.

Some researchers use foreground masks to perform recognition using sequences of silhouette images [9, 14]. A challenge with silhouette approaches is the requirement of accurate foreground/background segmentation, which is an open challenge in non-trivial settings.

Another approach is to develop a similarity measure between space-time volumes around a tracked subject. One example is the development of 3D correlation filters for matching volumes to action exemplars, as in [16]. Another example is the mapping of video volumes to a set of matrix manifolds where a manifold geodesic distance can be applied, as in [11]. Having to localize the action volume prior to recognition may seem to be a disadvantage, but evidence suggests that feature-based methods require localization to perform well on data sets that have complex and dynamic backgrounds [17], and bounding-box style localization is less onerous than silhouette extraction or body-part localization.

Our intent is to develop accurate action recognition methods that avoid overly-complex designs and can scale to large, real-world problems. We believe that by capturing latent geometric information in data sets, relatively simple manifold representations can provide powerful discriminative properties. A question is how to scale a nearest-neighbor manifold similarity measure to large data sets without losing accuracy.

In this paper we present a novel structure, called a Subspace Forest, which provides an efficient randomized-forest-based approximate nearest-neighbor mechanism for subspaces represented as points on Grassmann manifolds. While the use of randomized forests for approximating nearest-neighbor computations is well-known, this work extends the concept to sorting subspaces, which have no unique global origin or global coordinate system. We apply the Subspace Forest structure for action recognition by considering video segments as points on Grassmann manifolds, as inspired by the work of Lui *et al*. [11]. The Subspace Forest demonstrates state-of-the-art accuracy, with a dramatic improvement in scalability and speed. Our method also provides for a simpler design and fewer parameters than most feature-based systems, and is built without supervision.

2. Background

We use the term *tracklet* to denote a sequence of thumbnail-sized images of a subject tracked over a few seconds time. Tracklets are represented as a 3-dimensional data cube with axes of width, height, and frame number, (x, y, f). All tracklets are scaled to be the same size. Tracklets can be flattened into matrices by unfolding along any of the three axes.

Let T_i represent tracklet i. Let T_i^k be the matrix representing the tensor unfolding of T_i along axis k. Then, using QR Factorization, we have the following.

$$T_i^k = Q_i^k R_i^k \tag{1}$$

Q_i^k is an orthogonal basis for T_i^k. The distance between two tracklets, T_1 and T_2, can be computed with respect to unfolding k, using the principal angles between the subspaces spanned by Q_1^k and Q_2^k.

$$\begin{aligned} {Q_1^k}^T Q_2^k &= U \Sigma V^T \\ \cos\Theta &= diag(\Sigma) \end{aligned} \tag{2}$$

In Eq. (2), the Singular Value Decomposition is used to compute the principal angles between the subspaces. The singular values, given by the diagonal entries of Σ, are the cosines of the principal angles.

A Grassmann Manifold, $Gr(r, n)$, is the set of all r-dimensional linear subspaces of the n-dimensional vector space V. Points on a Grassmann Manifold are subspaces, and assuming the underlying field is $\Re$, can be identified by orthogonal matrices. By computing the orthogonal decomposition of the flattened matrix, one can identify the point on the Grassmann associated with the flattened data cube. In this manner, we map tracklets to points on Grassmann Manifolds, each tracklet being represented by the three subspaces arising from the three tensor unfoldings.

The chordal distance can be used to represent the geodesic distance between points on a Grassmann Manifold. This is a pair-wise comparison between two tracklets using the principal angles between the subspaces. The chordal distance, $d(T_1^k, T_2^k)$ between tracklets T_1 and T_2 along axis k is the L_2 norm of the component-wise sine function applied to the entries of Θ.

$$d(T_1^k, T_2^k) = \|\sin\Theta\|_2 \tag{3}$$

The scalability challenge is to perform nearest-neighbor based classification or clustering using this distance metric between subspaces without having to compute an all-pairs comparison, which is computationally intractible for large data sets. In addressing this challenge, we developed an approximate nearest neighbor structure that can be efficiently built, provides fast run-time nearest-neighbor queries, and respects the manifold geometry of the Grassmann representation by using the chordal distance between points. We call this structure the Subspace Forest because it is based on the application of randomized forests to sorting subspaces of a fixed dimension. Before presenting the details, we first provide background on other approximate nearest-neighbor methods.

2.1. Approximate Nearest Neighbors

The use of randomized forests for approximate nearest neighbors (ANN) is not new. One of the earlier works is a discussion by T. K. Ho on constructing decision forests using random subspace selection [6]. Ho's method involves projecting feature vectors to reduced-dimension subspaces by selecting a subset of the vector components for each tree in the forest. Ho evaluates this general concept using a number of different splitting strategies, including both supervised and unsupervised methods. This approach is a conceptual forerunner of the Subspace Forest, but the critical difference is that Ho generates decision trees off feature vectors that live in a space with a global origin, and thus the subspaces formed using random selection of vector components is easily partitioned using standard techniques.

Very similar to Ho's work, randomized forests of K-D Trees have become popular tools for scalable image retrieval [12, 19, 15] using Bag of Features representations. A popular implementation is the Fast Library for Approximate Nearest Neighbors (FLANN), developed by Muja and Lowe [13]. Whether sorting histograms, as with Bag of Features, or other feature vectors, FLANN and other ANN implementations based on randomized K-D Trees work by stochastically selecting a vector component with large variance to use in partitioning the feature space. As with Ho's approach of selecting a subset of features, this method requires a global coordinate system that is applied to all the data samples.

The following point is important to understanding a key contribution of this paper. Our representation of actions as points on Grassmann manifolds does not directly provide a global coordinate system for the samples. Distances are computed pair-wise, and Grassmann manifolds have no unique origin. Conceptually, one could map all the samples to the tangent space of an arbitrary point, and then apply FLANN in the resulting Euclidean space, but this approach is only valid if all the samples are sufficiently close to the arbitrary origin. The farther away the points are on the manifold, the worse the tangent-space approximation becomes, and the more error in the distance computations. The Subspace Forest structure we introduce avoids this issue by using computations that respect the manifold geometry.

2.2. Approximate Nearest Neighbors on Manifolds

Turaga *et al*. describes a general method for the application of existing nearest-neighbor search algorithms to non-Euclidean manifolds [20]. This paper is a good example of the approach outlined above, where points are mapped from the manifold onto a tangent space, and then a traditional ANN algorithm is applied. The mapping from the manifold to a tangent space is defined at a specific point on the manifold called the "pole", which forms the origin in the tangent space. Any point can serve as the pole, but one should select this point carefully because the mapping from the manifold to the tangent space is most accurate (in terms of preserving distances) for points closest to the pole. One mechanism to select the pole is by computing the Karcher Mean [7] of the data set, which is a computationally expensive iterative process. Our approach is to avoid the errors induced by mapping points onto tangent planes by using geodesic distances in our randomized forest structure.

Wang *et al*. present a method for developing spatial hashing functions for high dimensional data [21]. Conceptually similar to Locality Sensitive Hashing (LSH), [4], Wang's Grassmann Hashing (GRASH) improves over LSH by selecting optimal subspaces for hashing using a Grassmann metric. A key step in this approach is the use of Linear Discriminant Analysis (LDA) to provide a set of subspace candidates. Unlike the Subspace Forest, GRASH requires supervision to develop the hashing functions, and requires a metric space in which LDA can be computed over the training samples.

The most similar work to the Subspace Forest is a randomized manifold forest developed by Bonde *et al*. for learning kernel hyperparameters of a set-to-set face recognition algorithm [1]. Bonde's method shares the general idea of constructing randomized forests using pair-wise principal angle computations at the interior nodes of a decision tree. In comparison, our independently-developed Subspace Forest uses no kernel projections, no iterations to select optimal node splitting parameters, and no supervision. Although different, Bonde's results on set-to-set face recognition provides additional support for the general utility of manifold-based randomized forests.

3. Method

Below, we describe the Subspace Forest and related data structures, illustrating the construction of a Subspace Forest and how to apply it for nearest-neighbor-based action recognition. To validate the benefit of the Subspace Forest, we evaluate its accuracy in comparison to contemporary methods on standard benchmark classification data sets. We investigate its scalability by demonstrating timing results on a much larger data set.

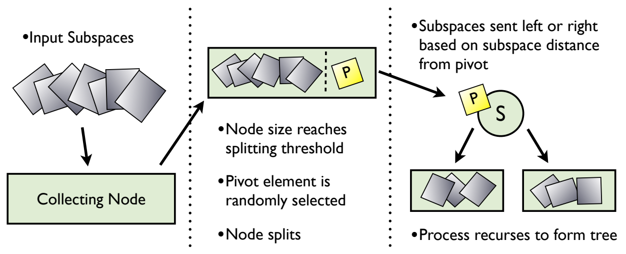
•Input Subspaces
Collecting Node
P
•Node size reaches splitting threshold
•Pivot element is randomly selected
•Node splits
•Subspaces sent left or right based on subspace distance from pivot
S
•Process recurses to form tree

Figure 1. Illustration of Subspace Tree construction.

3.1. Subspace Tree

We define a Subspace Tree as a tree structure used to sort points on a Grassmann manifold, which define subspaces of a fixed dimension. In principle, this structure could be used to sort any set of orthogonal matrices of fixed dimension. In our application, we use it to sort tracklets which have been flattened from a data cube into a matrix, from which an orthogonal basis is computed.

The basic formulation of a Subspace Tree (SSTree) is illustrated in Figure 1 and outlined in Algorithm 1. Samples are added to an initially empty node until it becomes large enough to consider splitting. We define τ to be the minimum number of elements in a node before it may be split. An element of the node being split is selected to be the pivot element. The chordal distance, Eq. (3), is computed between each element of the node and the pivot. Those having a distance less than or equal to a threshold are added to the left child node, the others to the right. The now-empty node is marked as a splitting node. This process recurses to form a tree, where all the samples are at leaf nodes, and all interior nodes are splitting nodes.

The SSTree imposes a local partial ordering at each splitting node, yet there is no global sort order. The tree works in a manner similar to randomized decision trees for generating approximate nearest neighbors, yet it operates on subspaces. By voting across a forest of SSTrees, we mitigate the errors inherent to any single tree that is constructed in this manner. The Subspace Forest constructed from a set of SSTrees is discussed in §3.3.

3.2. Subspace Tree Variations

In Algorithm 1, lines 5, 6, and 7 refer to functions that may implement different strategies for determining when and how to split a node in an SSTree. In this section, we discuss two node splitting variants called Median Splitting and Entropy Splitting. We also present a variation of the SSTree called the Random Axis Subspace Tree.

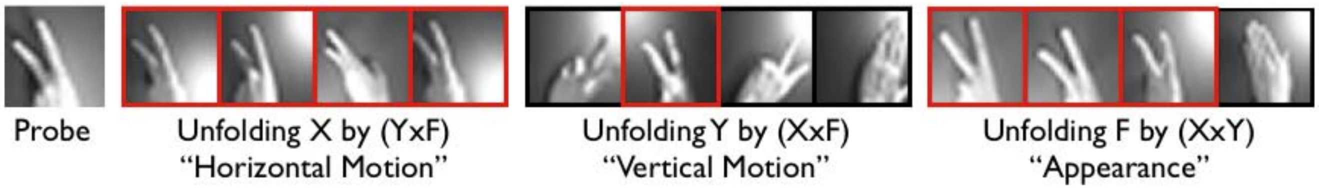

Figure 2. Top 4 neighbors, sorted left to right, from three trees of a Subspace Forest. Those matching the probe class are outlined in red. Samples are from the Cambridge Gestures data set [8]. The "Vertical Motion" unfolding, represented by the second tree, cannot distinguish left-moving from right-moving gestures.

Algorithm 1: Subspace Tree Construction

```
Data: $Node.items == \emptyset$
Data: $Node.status == Collecting$
Input: Orthogonal matrix $X$
begin
    if $Node.status == Collecting$ then
        $Node.items \leftarrow Node.items \cup X$
        if $|Node.items| > \tau$ then
5           $Node.pivot \leftarrow selectPivot()$
6           if $checkSplittingCriteria()$ then
7               $Node.threshold \leftarrow selectThreshold()$
                $Node.Left \leftarrow new(Node)$
                $Node.Right \leftarrow new(Node)$
                for $x_i \in Node.items$ do
                    $D_i \leftarrow d(x_i, Node.pivot)$        //Eq. (3)
                    if $D_i \leq Node.threshold$ then
                        $Node.Left.add(x_i)$
                    else
                        $Node.Right.add(x_i)$
                $Node.status \leftarrow Splitting$
                $Node.items \leftarrow \emptyset$
    else
        $D \leftarrow d(X, Node.pivot)$        //Eq. (3)
        if $D \leq Node.threshold$ then
            $Node.Left.add(X)$
        else
            $Node.Right.add(X)$
```

3.2.1 Median Splitting

With the median splitting strategy, the node is split as soon as the number of items in a node exceeds τ. There is no additional selection criteria, so the function of line 6 always evaluates as true. Pivot selection (line 5) is a random selection of one of the samples in the node. Threshold selection (line 7) is done by computing the median chordal distance between the pivot element and the items in the node. No labels are considered during the construction of this tree.

3.2.2 Entropy Splitting

With Entropy Splitting, the splitting criteria (line 6) is based on the distribution of the distances between the elements of the node and the selected pivot. The pivot is chosen randomly as with Median Splitting. Entropy is computed over a normalized histogram with a fixed number of bins covering an appropriate range of distances. Eq. (4) shows the entropy computation, where H is entropy, k is the number of histogram bins, p_i is the proportion of samples in bin i.

$$H = -\sum_{i=1}^{k} p_i \log p_i \tag{4}$$

When the entropy falls below an empirically-determined threshold, H_t, this indicates that the distribution has become concentrated in a small number of bins. Entropy is maximized by a uniform distribution and is zero when all samples are within a single bin. Threshold selection (line 7) is as follows. When the entropy falls below H_t, the distances are divided into two clusters and the midpoint between the cluster centers is used as the splitting threshold. Assuming τ is small, the entropy computation and clustering of the scalar distances adds a negligible amount of computational overhead in comparison to the median splitting strategy. No labels are considered during the construction of this tree. Distinct from supervised decision trees, our entropy computation is based on the distribution of distances, not distribution of labels.

3.2.3 Random Axis SSTree

In the previous discussion, we assumed the input to the tree was an orthogonal matrix, where the unfolding axis was selected a-priori and used for all samples. A forest can have trees representing each of the axes, but any single tree will represent only one axis. The Random Axis SSTree (RA-SSTree) takes a complete tracklet data cube as input, and sorts it per Algorithm 1, but with each new child node randomly selecting an unfolding axis. For the experiments reported on in this paper, we use Median Splitting for all RA-SSTrees.

3.3. Subspace Forest

The Subspace Forest consists of a set of SSTrees. When applied to action recognition, we select forest sizes in multiples of three so that we can have an equal number of

SSTrees for each unfolding axis of the tracklets. With RA-SSTrees, any number of trees can be used because each tree has information about all unfoldings. For classification, a video clip is flattened into three factor matrices, one each along the X,Y, and T dimensions. We compute an orthogonal basis for each of the three factors. Each of the three orthogonalized unfoldings of the input video are presented to corresponding subspace trees in the forest to determine the approximate nearest neighbors for that factor. To classify a probe sample, the label of the K-Nearest-Neighbors (KNN) from each leaf node is used in simple majority voting. Other voting strategies could be applied, for example soft weighting, but we do not explore them in this paper.

Figure 2 shows an example nearest-neighbor query result from a small forest of 3 Median Splitting SSTrees. The input sample is a hand gesture from the Cambridge Gestures data set (see §4.1.3), and the top matches from each tree are shown. Those outlined in red have the same class as the probe sample. This figure helps illustrate the benefit of including all three tracklet unfoldings in the forest. For different classes of tracklets, different unfoldings may contain the most discriminative information.

3.4. Tree and Parameter Selection

We compared the performance of Subspace Forests using different SSTree variants. We use the KTH Actions data set (see §4.1.1) for comparing tree variants. We varied the number of trees in the forest and ran 10 trials for each combination of variant and size. The results are shown in Figure 3. Overall, we felt that the Entropy Splitting strategy performed better than the other two. The best result was 97.9% when using Entropy Splitting and 27 trees in the forest. The RA-SSTree variant performed best when the forest size was limited to fewer than 9 trees, and yields competitive results (95.5%) even when limited to only 3 trees.

Beyond tree selection, there are only a few parameters required with a Subspace Forest: the size of the data cube to represent the action, the number of trees in the forest, N, and the number of samples a node will collect before splitting, τ. When using the entropy splitting strategy, H_t must also be selected.

Empirically, we found that a wide range of values for these parameters yields good results. We use the same size and type of Subspace Forest, with the same node splitting and entropy thresholds applied to all three data sets presented in our results. The only difference in application was the selection of an appropriate cube size based on the source video resolution and temporal extent of the actions. Specifically, we used a Subspace Forest of 27 Entropy Splitting trees with τ=21, and H_t=2.19.

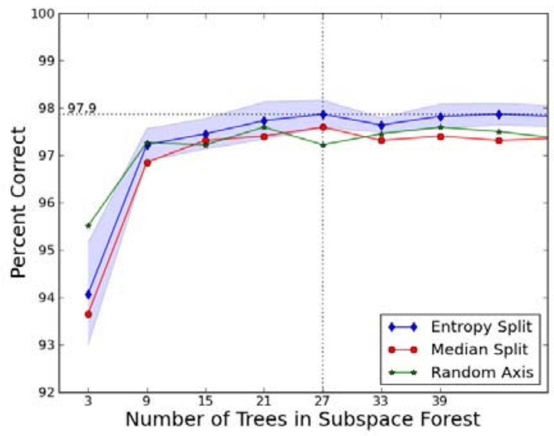

Figure 3. Accuracy vs. Forest Size for different tree types on KTH Actions. Each data point represents the mean of ten trials. The shaded region represents the 95% confidence interval around the mean values of the Entropy Splitting results. For large enough forests, we found that Entropy Splitting slightly out-performs the other variants. The Random Axis variant has the best performance when only a few trees are used.

4. Results

In this section, we present our results on Subspace Forest classification accuracy and scalability.

4.1. Classification Accuracy

We evaluate the Subspace Forest using three publicly available data sets: KTH Actions [18], UCF Sports [16], and Cambridge Gestures [8].

4.1.1 KTH Actions

KTH is one of the most well-known action recognition benchmarks, consisting of 25 subjects performing six actions (walking, jogging, running, boxing, clapping, waving) in four different scenarios, for a total of 600 videos.[1] To extract tracklets from KTH, we select a fixed position space-time bounding box, based on annotations provided by Lui [11], available from him upon request. The spatial extents of the bounding box is resized to 20x20 pixels, and we use 32 frames for the temporal window. Some actions do not have 32 consecutive frames of the person on-screen. In this case, we use a smaller temporal window and repeat frames as required to fill 32 frames.

We followed the training/testing split outlined by [18]. The Subspace Forest exceeds known state-of-the-art performance, as shown in Table 1. With our single-threaded python implementation, the total time to build the Subspace

[1]Technically, there are 599 KTH Action videos, as Person 13/Scenario3/Clapping is missing from the data set. We use a different section of Person 13/Scenario2/Clapping to replace it.

Forest on the training partition and recognize all the actions in the test partition is 6.5 minutes.

Method	**Accuracy**
Subspace Forest	**97.9**
CMOF [5]	97.4
Product Manifold [11]	96
MSTF [3]	94.5

Table 1. Comparison of the Subspace Forest to state-of-the-art methods on KTH Actions. All results in table use Schuldt's training/testing partitioning of the data. CMOF is Covariance Manifolds of Optical Flow, and MSTF is Mined Space-Time Features.

We believe that the Subspace Forest demonstrates both the highest accuracy and fastest end-to-end performance among published methods evaluated on KTH Actions using Schuldt's partitioning. It is hard to draw strong conclusions on the significance of the difference in accuracy among top methods, given statistical ceiling effects and minor implementation variations, yet our method is clearly a top-performer with the additional practical benefits of simplicity, speed, and scalability.

	walk	jog	run	box	clap	wave
walk	100	0	0	0	0	0
jog	0	100	0	0	0	0
run	0.6	0	99.4	0	0	0
box	0	0	0	100	0	0
clap	0	0	0	0	100	0
wave	0	0	0	0	12.0	88.0

Table 2. KTH Actions confusion matrix. Overall accuracy 97.9%.

Table 2 shows the confusion matrix. The Subspace Forest has little confusion between the similar classes of walking, jogging, and running, whereas it has some errors separating waving from clapping. This is in contrast to many other algorithms, and what might be expected of human judgment. For example, some of the subjects exhibit running in a way that is understandably confusing with jogging, yet no human would confuse any of the waving samples with clapping. This leads to the speculation that a combination of representations might further improve performance.

4.1.2 UCF Sports

UCF Sports is a relatively small data set, with 150 video clips representing ten actions: diving, swinging a golf club, kicking, weight lifting, horseback riding, running, skateboarding, swinging on a pommel horse, swinging on high bars, and walking. UCF Sports is challenging due to unequal set sizes, significant intra-class variability, and complex backgrounds. We use 64-frame tracklets of 32x32 pixels, selected from the middle frames of the samples, and using the bounding boxes provided by the data set, except for 10 videos where the bounds are not provided. For those 10, we created the associated tracks, and this additional data is available from us upon request. As with KTH, frames are repeated for samples with too few frames.

We followed the standard protocol for UCF Sports, which is leave-one-out classification. With the small size of this data set, even using leave-one-out, there are only 149 samples for use in constructing the Subspace Forest, so we doubled the gallery size by adding horizontally-mirrored copies of each training tracklet. Our results are as good as existing state-of-the-art methods, as shown in Table 3. The confusion matrix in Table 4 shows that many of our errors were from confusing skateboarding with walking.

Method	**Accuracy**
Subspace Forest	**91.3**
AFMKL [22]	**91.3**
Tangent Bundle [10]	88
DTM [2]	86.9

Table 3. Comparison of the Subspace Forest to state-of-the-art methods on UCF Sports. AFMKL is Augmented Feature Multi-Kernel Learning and DTM is Discriminative Topics Modelling.

	DV	GS	K	WL	HR	R	SK	PH	HB	W
DV	100	0	0	0	0	0	0	0	0	0
GS	0	94.4	0	5.6	0	0	0	0	0	0
K	0	0	100	0	0	0	0	0	0	0
WL	0	0	0	100	0	0	0	0	0	0
HR	0	8.3	0	0	91.7	0	0	0	0	0
R	0	0	0	0	7.7	84.6	0	0	0	7.7
SK	0	0	0	0	0	0	58.3	0	0	41.7
PH	0	0	0	0	0	0	0	100	0	0
HB	0	0	0	0	0	0	0	0	100	0
W	0	4.5	0	4.5	0	0	9.1	0	0	81.8

Table 4. UCF Sports confusion matrix. Overall accuracy 91.3%. Actions are: DV=diving, GS=golf swing, K=kicking, WL=weight lifting, HR=horseback riding, R=running, SK=skateboarding, PH=pommel horse swinging, HB=high bar swinging, and W=walking.

4.1.3 Cambridge Gestures

Cambridge Gestures are videos of hands in one of three shapes (flat, spread, v-shape) performing one of three motions (left, right, contraction), yielding 9 classes. There are 5 different sets, 20 samples per class per set, for a total of 900 videos. We select the middle 32 frames from each sample, and rescale the resolution to 20x20 pixels.

For classifying the gestures, we use the training/testing split outlined by Kim *et al.* [8], which requires training on Set 5 (180 samples), and testing on Sets 1-4. Our results are competitive with leading published results.

4.2. Scalability

Determining the nearest neighbors using an SSTree requires $D+\tau-1$ distance computations, D decisions to determine the leaf node, and then an additional $\tau-1$ computations, at maximum, to determine the nearest within the leaf

Method	**Accuracy**
Subspace Forest	89.8
Tangent Bundle [10]	**91**
TCCA [8]	82

Table 5. Classification accuracy on Cambridge Gestures using protocol from Kim *et al*. TCCA is Tensor Canonical Correlation Analysis.

node. Each chordal distance computation takes about 10 milliseconds using a video cube of dimensions (20x20x32). For example, with a tree constructed from 2,500 gallery samples, the expected depth is $\log(2500) = 11.3$, and with $\tau = 25$, the time to select the approximate nearest neighbor is about 0.36 seconds. Note that this is the time to recognize an entire tracklet, not to process a single frame. With 32-frame video clips, our *effective frame rate is about 88 fps*. Not only that, but this structure is highly scalable due to the $O(\log N)$ time complexity. Increasing from 2,500 to 10,000 gallery videos, the effective frame rate drops to 84 fps, and with 1,000,000 the recognition frame rate would remain at a respectable 71 fps. Tree construction is $O(N \log N)$. The Subspace Forest has the same time complexity as the SSTree, with a constant factor per tree assuming a serial implementation. Forests can be trivially parallelized by having a computational node per tree, which would allow for very efficient build and recognition times for large scale data sets. For the experiments described in this paper, we used a serial, single-threaded, python implementation on commodity hardware.

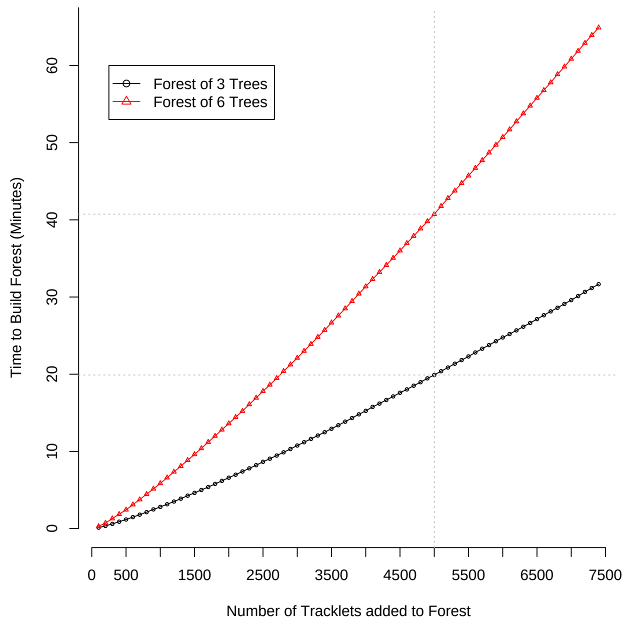

Figure 4. Subspace Forest construction time compared to the size of the training data.

Figure 4 demonstrates actual run-times required to generate Subspace Forests of different sizes. The figure shows how long it takes to construct a Subspace Forest, in minutes, in relation to the number of samples in the training data. The two lines represent Subspace Forests of 3 and 6 trees, using Entropy Splitting, $\tau = 21$, $H_t = 2.25$. Tracklet dimensions are (32x32x32). The slopes of the lines show a slight super-linear growth, which agrees with the $O(N \log N)$ complexity analysis. The light gray guide lines show that as the number of trees in the forest doubles, the construction time with our serial implementation also doubles from 20 to 40 minutes. A larger forest of 27 trees would take 9 times longer to build than a 3-tree forest with our serial implementation, but this additional time is easily mitigated with parallelism.

5. Conclusion

Lui *et al*. [11] showed that videos of human actions can be represented as points on Grassmann manifolds, and that the geodesic distances between two videos represented this way is an effective method for determining their similarity. In this paper, we introduced the Subspace Forest for determining the approximate nearest neighbors of points on Grassmann manifolds, which can be used to rapidly classify actions based on a set of labeled gallery samples. The Subspace Forest works in a manner conceptually similar to randomized forests for approximating nearest neighbor computations, but respects the Grassmann manifold geometry. With the Subspace Forest, action recognition can be performed with state-of-the-art accuracy and with the ability to scale to much larger systems than is feasible with many previously published approaches.

Other advantages of our method for action recognition include the simplicity of implementation [2], and the avoidance of the many design decisions and parameter selections that are required by bag-of-features methods. We build the Subspace Forest without using labels, so it is well-suited for unsupervised learning applications.

6. Acknowledgements

Thank you to Dr. Yui Man Lui for many insightful discussions. Research was supported in part by DARPA Contract W911NF-10-2-0066.

References

[1] U. Bonde, T. K. Kim, and K. Ramakrishnan. Randomised manifold forests for principal angle-based face recognition. In *Proc. Asian Conference on Computer Vision (ACCV)*, 2010. 3

[2] M. Bregonzio, J. Li, S. Gong, and T. Xiang. Discriminative topics modelling for action feature selection and recognition.

[2]Source code available upon request. Please contact primary author.

In *Proc. British Machine Vision Conference (BMVC)*, 2010. 1, 6

[3] A. Gilbert, J. Illingworth, and R. Bowden. Fast realistic multi-action recognition using mined dense spatio-temporal features. In *Proc. International Conference on Computer Vision (ICCV)*, 2009. 6

[4] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529. Morgan Kaufmann Publishers Inc., 1999. 3

[5] K. Guo, P. Ishwar, and J. Konrad. Action recognition using sparse representation on covariance manifolds of optical flow. In *Proc. IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 2010. 6

[6] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998. 2

[7] H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30(5):509–541, 1977. 3

[8] T. K. Kim, S. F. Wong, and R. Cipolla. Tensor canonical correlation analysis for action classification. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2007. 4, 5, 6, 7

[9] Z. Lin, Z. Jiang, and L. S. Davis. Recognizing actions by shape-motion prototype trees. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009. 1

[10] Y. M. Lui and J. R. Beveridge. Tangent bundle for human action recognition. In *Proc. IEEE Conference on Automatic Face and Gesture Recognition (FG2011)*, 2011. 6, 7

[11] Y. M. Lui, J. R. Beveridge, and M. Kirby. Action classification on product manifolds. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2010. 1, 5, 6, 7

[12] F. Moosmann, W. Triggs, and F. Jurie. Randomized clustering forests for building fast and discriminative visual vocabularies. In *Proc. Conference on Neural Information Processing Systems (NIPS)*, 2006. 2

[13] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proc. International Conference on Computer Vision Theory and Application (VISSAPP)*, 2009. 2

[14] F. Nater, H. Grabner, and L. Van Gool. Exploiting simple hierarchies for unsupervised human behavior analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 1

[15] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2007. 2

[16] M. D. Rodriguez, J. Ahmed, and M. Shah. Action MACH a spatio-temporal maximum average correlation height filter for action recognition. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2008. 1, 5

[17] M. S. Ryoo, C. C. Chen, J. K. Aggarwal, and A. Roy-Chowdhury. An overview of contest on semantic description of human activities (SDHA) 2010. In *Proc. International Conference on Pattern Recognition (ICPR)*, 2010. 1

[18] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *Proc. International Conference on Pattern Recognition (ICPR)*, 2004. 1, 5

[19] C. Silpa-Anan and R. Hartley. Optimised KD-trees for fast image descriptor matching. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2008. 2

[20] P. Turaga and R. Chellappa. Nearest-neighbor search algorithms on non-Euclidean manifolds for computer vision applications. In *Proc. Indian Conference on Computer Vision, Graphics and Image Processing*, 2010. 3

[21] X. Wang, Z. Li, L. Zhang, and J. Yuan. Grassmann hashing for approximate nearest neighbor search in high dimensional space. In *Proc. International Conference on Multimedia and Expo (ICME)*, 2011. 3

[22] X. Wu, D. Xu, L. Duan, and J. Luo. Action recognition using context and appearance distribution features. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2011. 6