MULTISCALE ONLINE TRACKING OF MANIFOLDS

Yao Xie, Jiaji Huang, Rebecca Willett

Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA

ABSTRACT

This paper describes a Multiscale Online Union of SubSpaces Estimation (MOUSSE) algorithm for online tracking of a time-varying manifold. MOUSSE uses linear subsets of low-dimensional hyperplanes to approximate a manifold embedded in a high-dimensional space. Each subset corresponds to the leaf node in a binary tree which encapsulates the multiresolution analysis underlying the proposed algorithm. The tree structure and parameters of the subsets are estimated and sequentially updated using a stream of noisy samples. For each update, MOUSSE requires only simple linear computations. The update of each hyperplane in the estimate is computed via gradient descent on the Grassmannian manifold. Numerical simulations demonstrate the strong performance of MOUSSE in tracking a time-varying manifold.

Index Terms— Multiscale analysis, online tracking, manifold learning, tree structure, low-dimensional approximation

1. INTRODUCTION

Many high-dimensional dynamical systems can be well approximated using low-dimensional structures. Applications where such low-dimensional structure can be used include computer network traffic [1], environmental monitoring [2], and video recognition and tracking [3]. In these applications, it is desirable to find a low-dimensional approximation to the data as well as a method to update the approximation and track the manifold dynamics.

A recent online algorithm called "Grassmannian Rank-One Update Subspace Estimation" (GROUSE) effectively tracks a single subspace using incomplete data vectors[2]. However, when the data are sampled from a manifold with non-negligible curvature, one linear subspace may not provide a good approximation. Recent work demonstrates that using a collection of several linear subspaces can improve approximation in the fixed sample setting. The Geometric Multi-Resolution Analysis (GMRA) wavelet decomposition was developed for analyzing intrinsically low-dimensional point clouds in high-dimensional spaces and successfully applied to a variety of real-world applications [4]. A related work[5] models data using Gaussian mixture model with low dimensional structure. However, these approaches are "batch" algorithms which relies on having all observations in memory when computing a low-dimensional approximation.

In contrast, this paper is focused on online estimation, which is important in a variety of settings. For instance, the volume of available data may be so large that batch methods which use all samples simultaneously are computationally infeasible. Alternatively, we may receive sequential samples from a dynamic environment and need to track the changing manifold structure in real-time. This paper describes the Multiscale Online Union of SubSpaces Estimation (MOUSSE) algorithm, which approximates and tracks a manifold using subsets lying on low-dimensional hyperplanes. There are four key challenges addressed by this work: (1) the update of each subspace must depend non-linearly on the location of the most recent observation relative to the current manifold estimate; (2) the number of subspaces needed for an accurate estimate which is robust to noise is unknown and may vary with time; (3) to operate efficiently online, the method must use as little memory and as few computational resources as possible; (4) when we refine our multiscale approximation of the manifold, we need accurate initialization of the newly formed subspaces to allow for fast adaptation to the manifold's dynamics. These challenges make the proposed approach a highly nontrivial extension of previous work.

In MOUSSE, the approximation is multiscale, and the multiscale structure is represented through a binary tree. The leaves of the tree are the subsets that are currently used for the approximation. Other nodes of the tree represent the subsets that can be used for different scales of approximation. The tree structure enables us to quickly adapt the manifold approximation both spatially (as a function of the manifold's local curvature) and temporally (as the manifold evolves). The parameters of the subsets are updated linearly, and the basis of each subset is updated using a variant of GROUSE. We show using a numerical example that MOUSSE can successfully track a one-dimensional curve embedded in a two-dimensional space as it changes over time. Our method is related to a probabilistic online manifold learning algorithm [3] [5], but uses simple non-parametric modeling and easy online updates that involve only linear computations. Furthermore, there are only modest memory requirements associated with our method, even as the number of samples grows.

2. MODEL

Suppose we are given a sequence of data $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t$, $t = 1, 2, \cdots$, $\boldsymbol{x}_t \in \mathbb{R}^D$, with D denoting the *ambient dimension*. The data are noisy measurements of a manifold

$$\boldsymbol{x}_t = \boldsymbol{v}_t + \boldsymbol{w}_t, \tag{1}$$

where $\boldsymbol{v}_t \in \mathcal{M}_t$, $\mathcal{M}_t \subset \mathbb{R}^D$, and the noise $\boldsymbol{w}_t$ is a zero mean white Gaussian random vector with covariance matrix $\sigma^2 \boldsymbol{I}_{D\times D}$. Our goal is to design an online algorithm that approximates the manifold using the samples received up to time t and updates the approximation with each new sample to track a slowly time-varying manifold.

We use a union of subsets to approximate the manifold $\mathcal{M}_t$: $\widehat{\mathcal{M}}_t = \bigcup_{n\in\mathcal{N}_t} \mathcal{S}_{n,t}$, where $\mathcal{N}_t$ contains indices of subsets that are used for approximation at time t, and $K_t \triangleq |\mathcal{N}_t|$ is the total number of subspaces for approximation at time t. Each of these subset lies on a low-dimensional hyperplane with dimension d and is parameterized as:

$$\begin{aligned}\mathcal{S}_{n,t} = \{\boldsymbol{v} \in \mathbb{R}^D : &\boldsymbol{v} = \boldsymbol{U}_{n,t}\boldsymbol{\beta} + \boldsymbol{c}_{n,t}, \\ &\boldsymbol{\beta}^\top \boldsymbol{\Lambda}_{n,t}^{-1}\boldsymbol{\beta} \le 1, \quad \boldsymbol{\beta} \in \mathbb{R}^d\},\end{aligned} \tag{2}$$

This work was supported by DARPA award # FA8650-11-1-7150, AFOSR award # FA9550-10-1-0390, AFOSR award # FA9550-11-1-0028, and NSF award # CCF-06-43947.

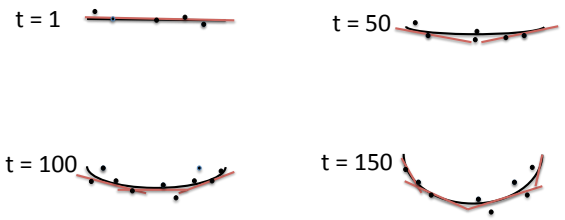


Fig. 1. Tracking of a time-varying manifold using multiscale low-dimensional subsets. The black curve is the underlying manifold, the dots are noisy samples from the manifold, and the red line segments are subset approximations.

where $\boldsymbol{U}_{n,t} \in \mathbb{R}^{D\times d}$ is a basis for the subspace and $\boldsymbol{c}_{n,t} \in \mathbb{R}^D$ is the offset of the hyperplane from the origin. The notation $\boldsymbol{U}^\top$ denotes transpose of a matrix $\boldsymbol{U}$. The diagonal matrix $\boldsymbol{\Lambda}_{n,t} \in \mathbb{R}^{d\times d}$ specifies the shape of a d-dimensional ellipsoid which reflects the local manifold curvature. The parameters above will be estimated online using streaming data. We assume $d \ll D$. Fig. 1 shows an example where $D = 2$ and $d = 1$, i.e., we use line segments to approximate and track a parabola with time-varying curvature. The black curve is the manifold, the black dots are the samples, and the red line segments are the subsets for approximation.

3. APPROXIMATE MAHALANOBIS DISTANCE

To update the subsets, whenever a new sample $\boldsymbol{x}_{t+1}$ becomes available, we have to determine the affinity of $\boldsymbol{x}_{t+1}$ to each subset. For this purpose, one candidate is the Euclidean distance of $\boldsymbol{x}_{t+1}$ to the hyperplane of each subset. However, this distance is problematic, because in our approximation the hyperplanes have boundaries defined by ellipsoids on these hyperplanes – a point can be close to a hyperplane but far away from the center of the subset. An alternative is the Mahalanobis distance, which takes into account the spread of the samples in the subset: $(\boldsymbol{x}_{t+1} - \boldsymbol{c}_{n,t})^\top \boldsymbol{\Phi}_{n,t}^{-1}(\boldsymbol{x}_{t+1} - \boldsymbol{c}_{n,t})$, where $\boldsymbol{\Phi}_{n,t}$ is the sample covariance matrix of the previous data used to estimate the n-th subset. However, this distance does not reflect the notion of low-dimensional subspaces since it does not depend on $\boldsymbol{U}_{n,t}$ and requires the storage of the $D \times D$ matrix $\boldsymbol{\Phi}_{n,t}$.

To address this challenge we introduce the following *approximate Mahalanobis distance*, which is a hybrid of the Mahalanobis distance and Euclidean distance. Define the projection of $\boldsymbol{x}_{t+1}$ onto the n-th hyperplane as

$$\boldsymbol{\beta} \triangleq \boldsymbol{U}_{n,t}^\top(\boldsymbol{x}_{t+1} - \boldsymbol{c}_{n,t}), \tag{3}$$

and the residual as

$$\boldsymbol{\beta}_\perp \triangleq (\boldsymbol{I} - \boldsymbol{U}_{n,t}\boldsymbol{U}_{n,t}^\top)(\boldsymbol{x}_{t+1} - \boldsymbol{c}_{n,t}). \tag{4}$$

The approximate Mahalanobis distance is given by

$$d(\boldsymbol{x}_{t+1}, \mathcal{S}_{n,t}) \triangleq \boldsymbol{\beta}^\top \boldsymbol{\Lambda}_{n,t}^{-1}\boldsymbol{\beta} + \delta_{n,t}^{-1}\|\boldsymbol{\beta}_\perp\|^2, \tag{5}$$

where $\delta_{n,t}$ is a parameter that estimates the magnitude of the fitting error per dimension along the $D-d$ dimensions orthogonal to the subspace, and $\|\boldsymbol{x}\|$ denotes the ℓ_2-norm of a vector $\boldsymbol{x}$.

The distance (5) is an approximation to the Mahalanobis distance in a low-dimension space. Indeed, if a data cluster has mean $\boldsymbol{c}$ and covariance matrix $\boldsymbol{\Phi}$, the Mahalanobis distance of new data $\boldsymbol{x}_{t+1}$ to the cluster is given by $(\boldsymbol{x}_{t+1} - \boldsymbol{c})^\top \boldsymbol{\Phi}^{-1}(\boldsymbol{x}_{t+1} - \boldsymbol{c})$. If we perform an eigendecomposition of $\boldsymbol{\Phi}$ and denote the largest d eigenvectors and eigenvalues as $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$, respectively, and the remaining $D-d$ eigenvalues and eigenvectors as $\boldsymbol{U}_\perp$ and $\boldsymbol{\Lambda}_\perp$, we can write the Mahalanobis distance as $(\boldsymbol{x}_{t+1} - \boldsymbol{c})^\top \boldsymbol{\Phi}^{-1}(\boldsymbol{x}_{t+1} - \boldsymbol{c}) = (\boldsymbol{x}_{t+1} - \boldsymbol{c}_t)^\top [\boldsymbol{U}\boldsymbol{\Lambda}^{-1}\boldsymbol{U}^\top + \boldsymbol{U}_\perp \boldsymbol{\Lambda}_\perp^{-1}\boldsymbol{U}_\perp^\top](\boldsymbol{x}_{t+1} - \boldsymbol{c}_t)$. Comparing this equation with (5), we see that the approximate Mahalanobis distance is equivalent to equating the minor eigenvalues to be a small constant $\boldsymbol{\Lambda}_\perp \approx \delta_{k,t}\boldsymbol{I}$, and $\boldsymbol{\Lambda}_{k,t} = \boldsymbol{\Lambda}$ contains the d largest eigenvalues of the covariance matrix.

4. MULTISCALE ONLINE UNION OF SUBSPACES ESTIMATION (MOUSSE)

In the following we describe Multiscale Online Union of Sub-Spaces Estimation (MOUSSE), which includes an initialization from training data and rank-one update using new data. MOUSSE uses a tree structure, which maintains the statistics necessary to enable subsequent multiscale updates. After initialization, when a new sample $\boldsymbol{x}_{t+1}$ is available, we update the approximation by first updating the parameters of all subsets (with updates weighted according to their distances to the new sample) and then updating the tree structure if necessary.

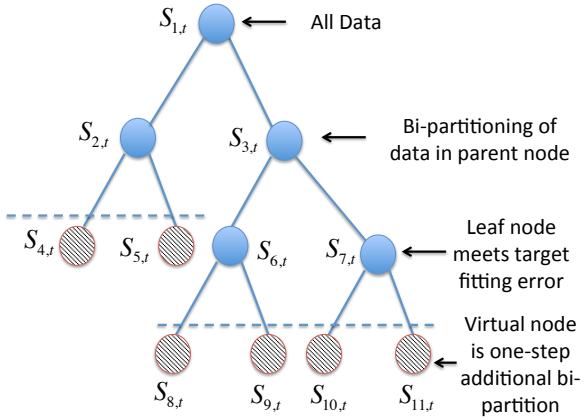


Fig. 2. Tree structure for subsets. The subsets for approximation is $\{\mathcal{S}_{2,t} \cup \mathcal{S}_{6,t} \cup \mathcal{S}_{7,t}\}$.

4.1. Tree Structure and Initialization

To initialize MOUSSE, we assume a small initial training set of samples, and perform a nested bi-partition of the training data set to form a tree structure, as shown in Fig. 2. The root of the tree represents the entire data set, the children of each node represent a bipartition of the data in the parent node. The bipartition of the data can be performed by the k-means algorithm. For each of these subsets, we compute the parameters for each node via Algorithm 1. The initialization is recursive: given a specified error tolerance $\epsilon = c\sigma$ for some constant c, typically with $c \in (1, 3)$, the bipartition continues until all leaf nodes have fitting error less than ϵ. Following this process, we perform one additional bi-partitioning of the data in the leaf nodes to form *virtual nodes*, which are subsets that are one-scale finer than the leaf nodes. The virtual nodes keep information we would use for a finer scale approximation. The parameters of the virtual nodes are initialized the same way as Step 2 to Step 5 in Algorithm 1. Finally the leaf nodes, i.e., the subsets $\mathcal{S}_{n,0}$ with indices $n \in \mathcal{N}_0$ are used as initial approximation to the manifold. The total number of nodes in the tree (including virtual nodes) is T_0, and the total number of subsets for

approximation (leaf nodes) is K_0. This tree construction is similar to that used in [4] except that here we introduce virtual nodes.

Algorithm 1 Initialize node n

Input: node index n, data $\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_{m_0}\}$ for n-th subset, tolerance ϵ
$\boldsymbol{c}_{n,0} := m_0^{-1} \sum_{m=1}^{m_0} \boldsymbol{y}_m$
Form covariance matrix:
$\boldsymbol{\Phi} := m_0^{-1} \sum_{m=1}^{n_0} (\boldsymbol{y}_m - \boldsymbol{c}_{n,0})(\boldsymbol{y}_m - \boldsymbol{c}_{n,0})^\top$
Form eigendecomposition:
$\boldsymbol{\Phi} = [\boldsymbol{U} \quad \boldsymbol{U}_\perp][\boldsymbol{\Lambda} \quad \boldsymbol{\Lambda}_\perp][\boldsymbol{U} \quad \boldsymbol{U}_\perp]^\top, \boldsymbol{U} \in \mathbb{R}^{D \times d}$
$\boldsymbol{U}_{n,0} := \boldsymbol{U}$, $\boldsymbol{\Lambda}_{n,0} := \boldsymbol{\Lambda}$, $\delta_{n,0} := \mathrm{tr}(\boldsymbol{\Lambda}_\perp)/(D-d)$
if $\delta_{n,0} > \epsilon$ **then**
  Bi-partition data
  Set two partitions as children, with index n_l and n_r
  Initialize node n_l and n_r
end if

4.2. Subset Update

We use the following method to update the subsets with new data. First we compute distance $d(\boldsymbol{x}_{t+1}, \mathcal{S}_{n,t})$, for all nodes in the tree $n \in \mathcal{T}_t$, where $\mathcal{T}_t$ contains indices for all nodes at time t. Then we linearly update the parameters of all the subsets using Algorithm 3. The step-size of update for each subset is inversely proportional to their distance to the new sample. In particular, we find the closest subset, denoted n_*, that has the minimum distance to the sample, denote this minimum distance as d_*, and denote its parent as n_p. Also find its virtual node child with smaller distance as n_v. Since all the ancestors of n_* are nested, they are the best approximations for the new sample at different scales. Hence we assign the same step-size $\alpha_0 > 0$ to n_*, n_v, and n_p and all the ancestor nodes. The step-size for other nodes are chosen to be exponentially decaying in distance. The step-size for the n-th subset is chosen to be

$$\alpha_{n,t} = \alpha_0 \exp\{-[d(\boldsymbol{x}_{t+1}, \mathcal{S}_{n,t}) - d_*]^2/(\gamma d_*^2)\}, \qquad (6)$$

where the constant $\gamma > 0$ controls the decay rate. We update the basis $\boldsymbol{U}_{n,t}$ for each node on the Grassmannian manifold using a modified version of GROUSE [2] for a hyperplane (instead for a subspace). The step-size η is chosen to be $\eta = \eta_0 \alpha_{n,t}/(\|\boldsymbol{x}_{t+1}\|\alpha_0)$, for a constant $\eta_0 > 0$.

4.3. Tree Structure Update

When the curvature of the manifold changes and cannot be accurately and parsimonisouly characterized by the current subset approximations, we update the tree structure by growing the tree or pruning the tree, which we refer to as "splitting" and "merging", respectively. Splitting increases resolution for approximation at the cost of higher complexity. Merging reduces resolution but lowers complexity. When making decision on splitting or merging, we have to take into considerations the approximation errors as well as the model complexity (the number of subsets K_t used in the approximation). This is related to the complexity-regularized tree estimation methods [6] and the notion of minimum description length (MDL) in compression theory. In particular, we use the sum of the fitting error plus a penalty on the number of subsets used for approximation as the cost function when deciding to split or merge. The regularization parameter $\mu > 0$ controls the tradeoff between the data fit and the estimator complexity. The complete steps of MOUSSE are summarized in Algorithm 2.

By splitting, we replace a leaf node with its two virtual nodes for approximation. For each of the new leaf node, we create two

Algorithm 2 MOUSSE

Input parameters: ϵ, μ
Initialize tree structure
for $t = 0, 1, \cdots$ **do**
  Given new data $\boldsymbol{x}_{t+1}$
  for $n \in \mathcal{N}_t$ **do**
    $\boldsymbol{\beta} = \boldsymbol{U}_{n,t}^\top (\boldsymbol{x}_{t+1} - \boldsymbol{c}_{n,t})$
    $\boldsymbol{\beta}_\perp = (\boldsymbol{I} - \boldsymbol{U}_{n,t}\boldsymbol{U}_{n,t}^\top)(\boldsymbol{x}_{t+1} - \boldsymbol{c}_{n,t})$
    $d(\boldsymbol{x}_t, \mathcal{S}_{n,t}) = \boldsymbol{\beta}^\top \boldsymbol{\Lambda}_{n,t}^{-1} \boldsymbol{\beta} + \delta_{n,t}^{-1} \|\boldsymbol{\beta}_\perp\|^2$
  end for
  Calculate $n_* = \arg\min_{n \in \mathcal{N}_t} d(\boldsymbol{x}_{t+1}, \mathcal{S}_{n,t})$
  Calculate step-sizes, update all nodes $n \in \mathcal{T}_t$
  if $\delta_{n_*,t} > \epsilon$ and
  $\delta_{n_v,t} + \mu \log(K_t + 1) < \delta_{n_*,t} + \mu \log(K_t)$ **then**
    Split n_*
  end if
  if $\delta_{n_*,t} < \epsilon$ and
  $\delta_{n_p,t} + \mu \log(K_t - 1) < \delta_{n_*,t} + \mu \log(K_t)$ **then**
    Merge n_* and its sibling
  end if
  Update $\mathcal{N}_t$ and $\mathcal{T}_t$
end for

Algorithm 3 Update node n

Input: node index n, $\alpha_{n,t}$, η, and subset parameters
Update: $\boldsymbol{c}_{n,t+1} = (1 - \alpha_{n,t})\boldsymbol{c}_{n,t} + \alpha_{n,t}\boldsymbol{x}_{t+1}$.
Update: $\lambda_{n,t+1}^{(m)} = (1 - \alpha_{n,t})\lambda_{n,t}^{(m)} + \alpha_{n,t}\beta_m^2$, $m = 1, \cdots, d$.
Update: $\delta_{n,t+1} = (1 - \alpha_{n,t})\delta_{n,t} + \|\boldsymbol{\beta}_\perp\|^2/(D-d)$
Update basis $\boldsymbol{U}_{n,t}$ using modified GROUSE
$\boldsymbol{r} = \boldsymbol{x}_{t+1} - \boldsymbol{c}_{n,t} - \boldsymbol{U}_{n,t}\boldsymbol{\beta}$
$\theta = \|\boldsymbol{r}\|\|\boldsymbol{U}_{n,t}\boldsymbol{\beta}\|$
$\boldsymbol{U}_{n,t} = \boldsymbol{U}_{n,t} + \frac{\cos(\theta\eta)-1}{\|\boldsymbol{\beta}\|^2}\boldsymbol{U}_{n,t}\boldsymbol{\beta}\boldsymbol{\beta}^\top + \sin(\theta\eta)\frac{\boldsymbol{r}}{\|\boldsymbol{r}\|}\frac{\boldsymbol{\beta}^\top}{\|\boldsymbol{\beta}\|}$.

new virtual nodes. Denote the index of a new leaf as n_0, and the pair of new virtual nodes as n_l and n_r. We initialize the parameter of n_l and n_r as

$$\begin{aligned} \boldsymbol{c}_{n_l,t} &= \boldsymbol{c}_{n,t} + \sqrt{\lambda_{n,t}^{(1)}}/2\boldsymbol{u}_1, \\ \boldsymbol{c}_{n_r,t} &= \boldsymbol{c}_{n,t} - \sqrt{\lambda_{n,t}^{(1)}}/2\boldsymbol{u}_1, \end{aligned} \qquad (7)$$

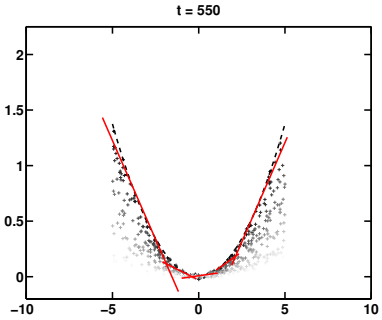
where $\boldsymbol{u}_1$ is the first column in $\boldsymbol{U}_{n_0,t}$. By doing so we split the subset of the parent node into two along the longest axis of the ellipsoid. The basis of n_l and n_r are $\boldsymbol{U}_{n_l,t} = \boldsymbol{U}_{n_r,t} = \boldsymbol{U}_{n_0,t}$, and the ellipsoid parameters of n_l and n_r are $\lambda_{n_l,t}^{(1)} = \lambda_{n_r,t}^{(1)} = \lambda_{n_0,t}^{(1)}/2$, and $\lambda_{n_l,t}^{(m)} = \lambda_{n_r,t}^{(m)} = \lambda_{n_0,t}^{(m)}$, $m = 2, \cdots, d$, respectively.

For merging, we replace two leaf nodes n_1 and n_2 with their parent node n_p, turn nodes n_1 and n_2 into virtual nodes, and delete the four virtual nodes of n_1 and n_2.

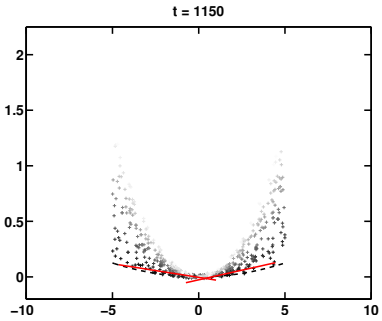
5. NUMERICAL EXAMPLE

We consider a time varying manifold in $\mathbb{R}^2$, with $D = 2$, $d = 1$. Points on the manifold $\boldsymbol{v}_t \triangleq [v_{t,1}, v_{t,2}]^\top$ obey $v_{t,2} = a(t)v_{t,1}^2$. The curvature $a(t) = at$, for $t = 1, \cdots, 600$, and $a(t) = a(1200 - t)$, for $t = 601, \cdots, 1200$. The rate of change is $a = 10^{-4}$, the error tolerance is given by $\epsilon = 10^{-3}$. The other parameter values are $\eta_0 = 0.1$, $\alpha_0 = 0.1$, $\gamma = 0.01$, and $\mu = 10^{-3}$. The data are generated by adding noise with variance $\sigma^2 = 10^{-4}$.

The noise variance is comparable to the minimum sample mean over t (which is 10^{-4}). Proof-of-concept results are in a video online at http://people.ee.duke.edu/~yx44/MOUSSE.m4v. In this display, the dashed line corresponds to the true manifold, the red lines correspond to the estimated union of subspaces, and the $+$ signs correspond to the past 500 samples, with darker colors corresponding to more recent observations. From this video, it is clear that we are effectively tracking the dynamics of the manifold, and keeping the representation parsimonious so the number of subspaces used by our model is proportional to the curvature of the manifold. As the curvature increases and decreases, the number of subspaces used in our approximation similarly increases and decreases. The snapshots of this video at time $t = 550$ and $t = 1150$ are shown in Fig. 3. The number of subsets K_t and fitting error as a function of time are shown in Fig. 4. The red line in Fig. 4(b) corresponds to 10^{-3}.



(a) t = 550

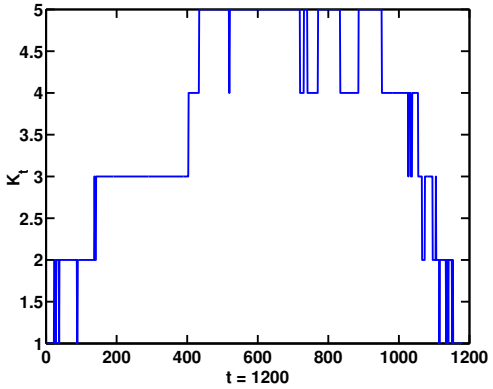


(b) t = 1150

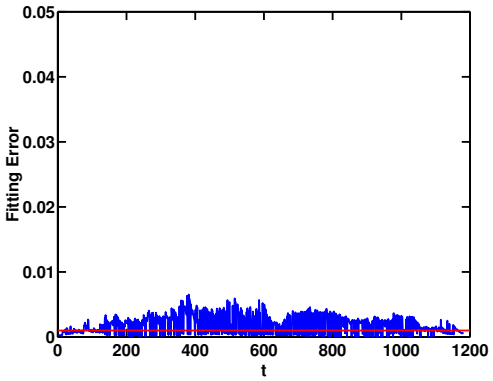
Fig. 3. Approximation of MOUSSE at $t = 550, 1150$.

6. CONCLUSIONS

This paper presents MOUSSE, which can approximate and track a time-varying manifold using subsets supported on low-dimensional hyperplanes. We have demonstrated the good performance of MOUSSE in tracking a curve in two-dimensional space. We anticipate our method will be robust to missing elements in each observation vector, and this is a key element of our ongoing work. Our method is very fast and scalable because of the multi-scale structure we employ. Moreover, our method does not need the knowledge of model order (i.e., the best number of subsets to use at each time) since our method is fully data adaptive. Our ongoing research include studying the performance of MOUSSE in approximating and tracking high-dimensional and experimental data.



(a) Number of subsets K_t



(b) Fitting error $\delta_{n_*,t}$

Fig. 4. Performance of MOUSSE.

7. REFERENCES

[1] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *Proc. of SIGCOMM*, 2004.

[2] L. Balzano, R. Nowak, and B. Recht, "Online identification and tracking of subspaces from highly incomplete information," in *Proceedings of the Allerton Conference on Communication, Control and Computing*, Sept. 2010, pp. 704 – 711.

[3] K.-C. Lee and D. Kriegman, "Online learning of probabilistic appearance manifolds for video-based recognition and tracking," in *Proceedings of CVPR*, 2005, pp. 852 – 859.

[4] W. Allard, G. Chen, and M. Maggioni, "Multi-scale geometric methods for data sets II: Geometric multi-resolution analysis," *Applied and Computational Harmonic Analysis*, vol. In press, 2011.

[5] M. Chen, J. Silva, J. Paisley, C. Wang, D. Dunson, and L. Carin, "Compressive sensing on manifolds using a nonparametric mixture of factor analyzers: algorithm and performance bounds," *IEEE Trans. on Signal Processing*, vol. 58, no. 12, pp. 6140 – 6155, 2010.

[6] L. Breiman, J. Friedman, R. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1983.