PATH-BASED SPECTRAL CLUSTERING FOR DECODING FAST TIME-VARYING MIMO CHANNELS

Steven Van Vaerenbergh[a], Ignacio Santamaría[a*], Paolo E. Barbano[b†], Umut Ozertem[c], Deniz Erdogmus[d ‡]*

[a]Department of Communications Engineering, University of Cantabria, Santander, Spain
[b]Department of Mathematics, Yale University, New Haven, Connecticut, USA
[c]Yahoo! Labs, Santa Clara, California, USA
[d]Department of Electrical and Computer Engineering, Northeastern University, Boston, Massachusetts, USA

ABSTRACT

In this paper, we present a clustering technique for decoding fast time-varying multiple-input multiple-output (MIMO) channels. The proposed method builds upon previous work that exploited the symmetry of the constellation and the order of the data within a spectral clustering procedure. The novelty of this work is that by adjusting the different steps of the standard spectral clustering algorithm, it introduces the expected shape of the clusters into the clustering process. The main modification applies to the construction of the weighted graph, for which it is shown that a path-based kernel, the *connectivity kernel*, can be a more appropriate similarity function than the Gaussian kernel. The obtained spectral clustering method is capable of finding clusters in sequential data. Experimental results are included to demonstrate the validity of the method.

1. INTRODUCTION

In the last decades, multiple-input multiple-output (MIMO) wireless communication technology has gained considerable attention, since it can offer significant increases in spectral efficiency compared to traditional single-input single-output (SISO) transmissions. A number of computationally efficient algorithms have been proposed for reliable symbol detection in flat-fading MIMO systems, based on the assumption that the MIMO channel is static and known at the receiver side. Nevertheless, their direct application in fast time-varying environments is difficult. A few adaptive equalization algorithms have been proposed that adaptively estimate the channel based on decision feedback [1, 2].

An alternative approach to equalization can be based on clustering, exploiting the fact that, for channels excited by signals belonging to a finite alphabet, the noisy observations tend to cluster around a finite number of *channel states*. Once these channel states have been estimated, channel equalization reduces to a classification problem. The main drawback of these algorithms is that the number of clusters grows exponentially as M^{N_t}, where M is the constellation size and N_t is the number of transmit antennas. Recently, methods have been proposed to reduce the number of clusters to estimate by exploiting the input constellation geometries [3, 4] and to extend from single-input to multiple-input systems [5]. However, these clustering methods consider only time-invariant environments.

In fast time-varying systems, the variations in the mixing matrix provoke a movement of the cluster centers and, consequently, the clusters adopt non-convex shapes and overlap each other. Conventional clustering algorithms such as k-means and expectation-maximization (EM) learning would fail for these non-convex clusters. In [6, 7] the authors presented a clustering technique that can deal with fast time-varying systems. By adding a temporal dimension to the scatter plot, this method converted the overlapping clusters into intertwined threads, which are then clustered using a standard form of spectral clustering. Moreover, the geometry of the constellation was taken into account to reduce the number of clusters to retrieve.

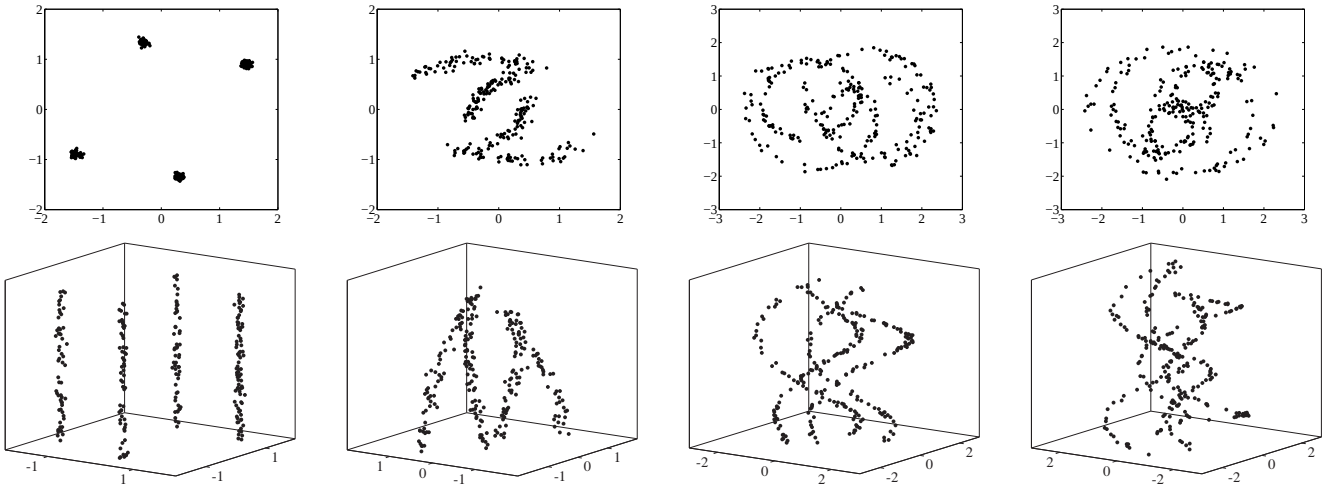
This paper aims to improve the previously presented technique by incorporating more information on the problem into the clustering procedure. First, the expected elongated shape of the clusters is taken into account by using a path-based similarity function, the *connectivity kernel*. Second, the order of the data is taken into account in the final clustering stage to avoid invalid solutions. Other contributions of this paper include the design of an efficient scheme to calculate the connectivity kernel for sequential data, and a procedure to select its kernel scale.

*This work was supported by MEC (Ministerio de Educación y Ciencia) under grant TEC2007-68020-C04-02 TCM (MultiMIMO), FPU grant AP2005-5366 and Consolider-Ingenio CSD 2008-00010 (COMONSENS).

†Partially sponsored by JIEDDO Grant No. W911NF0810020.

‡Supported by NSF grants ECS-0524835, ECS-0622239.

E-mail: {steven,nacho}@gtas.dicom.unican.es, paoloemilio.barbano @yale.edu, umut@yahoo-inc.com, erdogmus@ece.neu.edu



(a) $f_dT = 0$ (b) $f_dT = 0.001$ (c) $f_dT = 0.005$ (d) $f_dT = 0.01$

Fig. 1. Effect of different normalized Doppler spreads on the received data symbols. Top row: Scatter plots of the data received by one receive antenna for a BPSK MIMO system with 2 transmit antennas, for different values of the normalized Doppler frequency f_dT. Bottom row: Scatter plots to which the time index was added as an additional vertical axis. Due to the channel changes during the transmission of the data block, curved threads appear in this plot.

2. PROBLEM STATEMENT

In a typical MIMO flat-fading system with N_t transmit and N_r receive antennas, the N_r-dimensional received vector $\mathbf{x}_n$ at time n is expressed as

$$\mathbf{x}_n = \mathbf{H}_n \mathbf{d}_n + \mathbf{v}_n \tag{1}$$

where $\mathbf{H}_n$ is the complex $N_r \times N_t$ channel matrix whose elements represent independent flat-fading SISO channels, $\mathbf{d}_n$ contains the N (in general, complex) symbols transmitted by the N_t antennas at time n, and $\mathbf{v}_n$ represents both spatially and temporally white complex zero-mean Gaussian noise. The goal of blind symbol decoding is to estimate the symbols $\mathbf{d}_n$ given only the received data points $\mathbf{x}_n$.

In MIMO systems with *fast time-varying* channels, the channel matrix changes from symbol to symbol due to the Doppler spread caused by the movement of the transmitter and/or receiver. In such systems, depending on the Doppler spread, the channel matrices $\mathbf{H}_n$ are temporally correlated.

The top row of Fig. 1 illustrates the effect of different Doppler spreads on the scatter plot of a received data block in a typical binary phase-shift keying (BPSK) MIMO system, for which the basic constellation points are $d \in \{+1, -1\}$. Whereas the received data in a static system contains clearly separable clusters (Fig. 1(a) top), in a time-varying system these will overlap (Fig. 1(b), (c) and (d) top) and classical clustering algorithms that operate directly on the data will fail.

3. SPECTRAL CLUSTERING-BASED APPROACH

3.1. Exploiting the data order and geometry

In [6] the authors noted that adding the temporal dimension to the received data "untangles" the overlapping clusters into intertwined threads, as can be seen in the bottom row of Fig. 1. Based on this insight, a clustering procedure was designed to retrieve the different threads. At the core of this procedure lies a spectral clustering algorithm, which is able to retrieve non-convex clusters based on pairwise similarities using a Gaussian kernel.

Additionally, to improve the performance of the decoding algorithm in cases where clusters were not densely populated or difficult to distinguish, some geometrical information was incorporated into the clustering algorithm: As can be seen in Fig. 1, the symmetry of the constellation translates into the clusters following symmetric trajectories. By treating symmetrical clusters as one in a first clustering stage, an easier clustering problem was obtained in which less but denser clusters were to be found. In a second stage, each group of symmetrical clusters was again divided to find the final clusters. Details can be found in [6, 7].

This technique obtained very satisfactory results. In cases of high noise or high Doppler spread, however, it did not exclude invalid solutions in which clusters were bifurcated or two threads were clustered as one. To avoid these situations, a restriction needs to be built into the clustering procedure to avoid clusters that do not consist of single threads, or clusters that do not approximately span the entire time range.

In the sequel, we will first give a general description of spectral clustering and then propose some modifications to retrieve clusters that have this specific shape.

3.2. Spectral Clustering

Consider the weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ derived from the data such that the vertices are the data points $\mathcal{V} = \{\mathbf{x}_i\}$ and where each edge (i, j) is assigned a weight that expresses the similarity between its two vertices. An often used similarity measure is the Gaussian kernel:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{d_{i,j}^2}{\sigma^2}\right), \tag{2}$$

where $d_{i,j}$ is some distance measure such as the Euclidian distance $d_{i,j} := \|\mathbf{x}_i - \mathbf{x}_j\|$ and σ is the *kernel size*. The affinity matrix $\mathbf{K}$ (also called *similarity* or *kernel* matrix) of a weighted graph is the matrix that contains the kernel functions $K_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ between all point pairs. Spectral clustering is performed by analyzing the spectrum of this matrix. One of the most successful spectral clustering algorithms is the Ng-Jordan-Weiss (NJW) algorithm, introduced in [8]. It can be summarized in the following three steps:

1. *Affinity matrix*: Obtain this matrix from a graph using $K_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and $K_{ii} = 0$, for $i, j = 1, \ldots, N$.

2. *Eigenvectors of the graph Laplacian* $\mathbf{L}$: Obtain $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{N} K_{ij}$. Form the matrix $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$ containing the m largest eigenvectors of $\mathbf{L}$, where m is the number of subsets to retrieve.

3. *Eigenvector clustering*: Treat the rows of $\mathbf{V}$ as points in $\mathbb{R}^m$, and normalize them to unit length. Cluster them with an algorithm such as k-means. Assign the original point $\mathbf{x}_i$ to cluster j if and only if row i of the matrix $\mathbf{V}$ was assigned to cluster j.

All spectral clustering algorithms follow a similar three-step procedure, but they differ in the details of each step [8, 9]. In the first step, a kernel function should be chosen that reflects the information about the problem as much as possible. For instance, if the goal is to retrieve dense clusters of connected points, the Gaussian kernel (2) can be used. In the second step, a "graph Laplacian" matrix is constructed and its eigenvectors are retrieved. This step can be interpreted as finding a low-dimensional representation of the data. Thanks to the properties of the graph Laplacian, the obtained points now form compact and well-separated clusters [9], which can be retrieved in the third step by a simpler clustering procedure such as k-means.

This kernel function is very useful for clustering dense cloud-like shapes of points. However, when the data points form thread-like shapes such as in Fig. 1 it would be convenient to use a kernel function that adjusts better to the shape of the data groups. In the following section a kernel function is designed that favors such shapes.

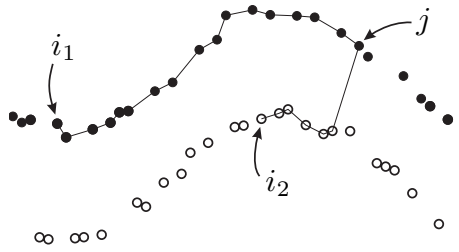


Fig. 2. Path-based similarity: Due to the optimal paths generally following dense regions of data, vertices i_1 and j are considered more similar than i_2 and j.

4. OPTIMIZING THE KERNEL FUNCTION

4.1. Path-based Spectral Clustering

Path-based clustering [10, 11] is a recently developed technique for clustering groups of points that are elongated in addition to being dense. In graph theory, a path p in a graph $\mathcal{G}$ is an alternating sequence of vertices and edges, beginning and ending with vertices, in which all vertices are distinct and each edge is incident with the vertex immediately preceding it and with the vertex immediately following it.

Let us denote by $\mathcal{P}_{i,j}$ the set of all paths from vertex i to vertex j. When dealing with elongated structures, two points should be considered similar if there is a clear path between them, in the sense that all of its edges are short. For instance, in Fig. 2, the vertices i_1 and j belong to the same cluster. This is reflected in the fact that there is a path between them that consists only of short edges. On the other hand, the vertices i_2 and j belong to different clusters, and any path between them will contain at least one longer edge.

Based on this observation, in [10] a kernel function was proposed that calculates the similarity between two vertices i and j based on the *weakest link of the best path* between them. The *weakest link* of a path p is considered to be its longest edge. We will denote the length of this edge as the "effective distance" of the path, which can be written as

$$\bar{d}_{i,j}^{p} = \max_{(k,l)\in p} d_{k,l}. \tag{3}$$

The *best path* between the two vertices will be the one whose effective distance is shortest, and we denote the *effective distance between the two vertices* as

$$\bar{d}_{i,j} = \min_{p\in\mathcal{P}_{i,j}} \bar{d}_{i,j}^{p}. \tag{4}$$

Based on this metric, the similarity between vertices i and j can be expressed using the Gaussian kernel as

$$\begin{aligned} \kappa_c(\mathbf{x}_i, \mathbf{x}_j) &= \exp\left(-\frac{\bar{d}_{i,j}^2}{\sigma^2}\right) \\ &= \exp\left(-\frac{1}{\sigma^2} \min_{p \in \mathcal{P}_{i,j}} \max_{(k,l) \in p} d_{k,l}\right). \end{aligned} \quad (5)$$

This kernel function is called "connectivity" kernel.

Although the computation of the kernel function (5) contains a min-max operation over all possible paths between two vertices, a recursive scheme can easily be applied to calculate the kernel function based on the result for shorter segments, for instance by using Dijkstra's shortest path algorithm [12]. In the following an efficient algorithm is presented for sequential data.

4.2. Adaptation for Sequential Data

The application considered in this paper deals with data symbols from communications, which are sent and received sequentially. While the previous section described the *general* path-based clustering method, the problem treated here deals with ordered data and therefore it will benefit from incorporating temporal information into the clustering process. Note that most kernel methods and clustering algorithms do not take into account any order in the incoming data.

The connectivity kernel can respect the data order by only considering paths that are "monotonic" in the temporal dimension. In other words, the paths used in kernel (5) should either consist *only* of edges $(k, l) \in p$ that fulfill $l > k$, or *only* of edges that fulfill $k > l$. Moreover, this restriction greatly reduces the total number of paths that need to be taken into account, which is very convenient for communications problems.

The following scheme describes how to efficiently calculate the effective distances between all pairs of data points, resulting in an effective distance matrix $\bar{\mathbf{D}}$. Once obtained, the similarity between points can be obtained by calculating (5). Let us denote by $\delta = j - i$ the "temporal separation" between points $\mathbf{x}_i$ and $\mathbf{x}_j$. We will fill the effective distance matrix $\bar{\mathbf{D}}$ one diagonal at a time, in an inductive manner:

1. $\delta = 0$: Elements on the diagonal of $\bar{\mathbf{D}}$ correspond to pairs of identical points, and therefore $\bar{d}_{i,i} = 0, \forall i$.

2. $\delta = 1$: Elements on the first upper diagonal are consecutive data points, for which $\bar{d}_{i,i+1} = d_{i,i+1}, \forall i$.

3. $\delta = l$: Elements on the l-th upper diagonal can be calculated based on the results obtained for $\delta < l$. The optimal path between $\mathbf{x}_i$ and $\mathbf{x}_j$ is either a direct connection of $\mathbf{x}_i$ and $\mathbf{x}_j$, or a combination of two shorter paths, as illustrated in Fig. 3. The effective distance of the direct connection is simply its Euclidian distance, while the effective distance of a combination of shorter paths can be calculated as the maximal effective distance of its parts. The resulting effective distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is the minimum of the effective distances over these paths.

4. $\delta < 0$: Due to symmetry of the kernel, $\bar{d}_{j,i} = \bar{d}_{i,j}$.

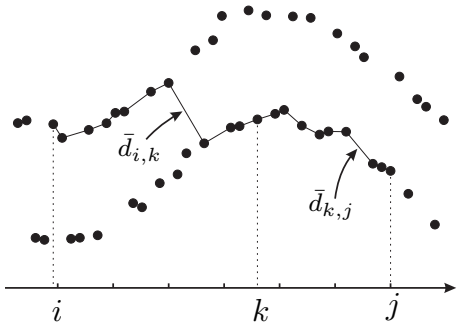


Fig. 3. The effective distance between vertices i and j can be calculated by considering all combinations of two shorter optimal paths, from i to k and from k to j, with $k = i, \ldots, j$.

4.3. A Self-Tuning Connectivity Kernel

Since the clustering procedure is very sensitive to the kernel width σ, the algorithm proposed in [6] used a "local scaling" procedure for the Gaussian kernel. Instead of using a single global σ for all data points, a local kernel width σ_i was assigned to each point, equal to the median of distances to its K-th nearest neighbors [13]. The same idea can be applied to increase the robustness of the connectivity kernel (5), leading to the following *locally scaled* connectivity kernel:

$$\kappa_{lc}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\bar{d}_{i,j}^2}{\sigma_i \sigma_j}\right). \quad (6)$$

This kernel is obtained in a straightforward manner by scaling each distance $d_{i,j}$ by the local scales σ_i and σ_j, before calculating the effective distance $\bar{d}_{i,j}$.

5. OPTIMIZING THE EIGENVECTOR CLUSTERING

Up till this point, the described algorithm favors elongated groups of points by making use of the connectivity kernel in its first step. In its second step, the eigenvectors of the graph Laplacian are retrieved, whose rows correspond to the input data points but allow for easier clustering. Optimally, the clustering in the last step should retrieve thread-like clusters that span as much of the time range as possible, if the temporal dimension is taken into account again. To rule out

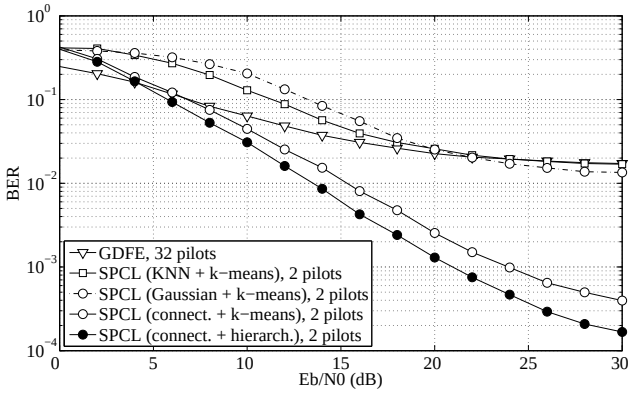


Fig. 4. Performance comparison of different algorithms for a 2×2 BPSK system with $f_d T = 0.005$.

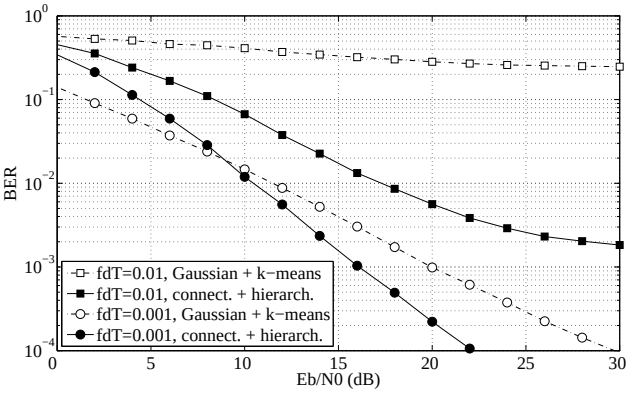


Fig. 5. Comparison of the initial [6] and the proposed algorithm at different Doppler spreads, for a 2×2 BPSK system.

invalid solutions such as bifurcated threads, which can occur by plainly applying k-means at this point, we propose to add the temporal dimension again to the obtained points and use hierarchical clustering to detect the final clusters.

In agglomerative hierarchical clustering, each point is initialized as a cluster. Subsequently, the two closest clusters are joined, and this process is repeated until the desired number of final clusters is obtained. The distance between clusters is measured as the minimal Euclidian distance between any node of the first cluster and any node of the second cluster, which is known as "single linkage" hierarchical clustering. The additional constraint we add to avoid invalid solutions, is that if two clusters overlap in time, they are only linked if at least $m - 1$ additional clusters overlap in the same time fraction. If this is not fulfilled the cluster pair is not merged but skipped in this iteration, and the next closest pair of clusters is considered.

6. TEST RESULTS AND COMPARISON

Computer simulations were carried out to illustrate the performance of the proposed algorithm. The following parameters were assumed: A BPSK signal was used, the channels were independent Rayleigh flat-fading and the temporal correlation of the channels was based on the Clarke and Gans Fading Model [14]. Each data block consisted of 256 slots (i.e. an $N_t \times 256$ matrix of BPSK symbols is transmitted). Different decoding algorithms were compared:

- The adaptive GDFE method from [1], using 32 pilot symbols and a forgetting factor λ of 0.95.
- Spectral clustering (SPCL) with a K-nearest neighbor (KNN) kernel, a standard Laplacian matrix, and k-means eigenvector clustering.
- The spectral clustering algorithm from [6], which uses a locally scaled Gaussian kernel, the Laplacian matrix from the NJW algorithm and k-means.
- The algorithm from [6] in which the Gaussian kernel was replaced by a locally scaled connectivity kernel.
- The proposed spectral clustering algorithm, which uses a locally scaled connectivity kernel, the Laplacian from the NJW algorithm and hierarchical clustering.

For the clustering algorithms, the optimal number of neighbors was determined experimentally as $K = 14$, and the scaling of the temporal axis was fixed as $t_n = 5 \cdot f_d T \cdot n$ with $n = 0, \ldots, 255$. Once the clusters were retrieved, the original symbols were decoded using only N_t pilot symbols (see [6]), placed in the middle of the frame.

In a first setup, a 2×2 system with a Doppler frequency of $f_d T = 0.005$ was considered. Fig. 4 shows the bit error (BER) rates for the compared algorithms. It can be observed that the algorithm from [6] (third curve) performs poorly for high noise levels, but reaches the same performance as the GDFE for low noise situations. When the Gaussian kernel is replaced by the connectivity kernel, a dramatic increase in performance is seen, as shown by the fourth curve. An additional improvement in performance can be achieved when moreover the final k-means clustering method is replaced by the described hierarchical clustering method.

In Fig. 5 the change in performance due to different normalized Doppler spreads is shown, for the spectral clustering algorithm from [6] and the proposed algorithm.

As a final example, Fig. 6 shows the clustering result on a data set that is contaminated by impulsive noise. The noise pdf of this example is $p(v) = 0.9p_1 + 0.1p_2$, where p_1 and p_2 are zero-mean Gaussian white noise distributions with $Eb/N0 = 15$dB and $Eb/N0 = -15$dB, respectively. If the outliers do not lie in between the clusters (as in Fig. 6), the connectivity kernel-based clustering is not disrupted when it retrieves 4 clusters. When 5 clusters are retrieved, the outliers are grouped as a new cluster.

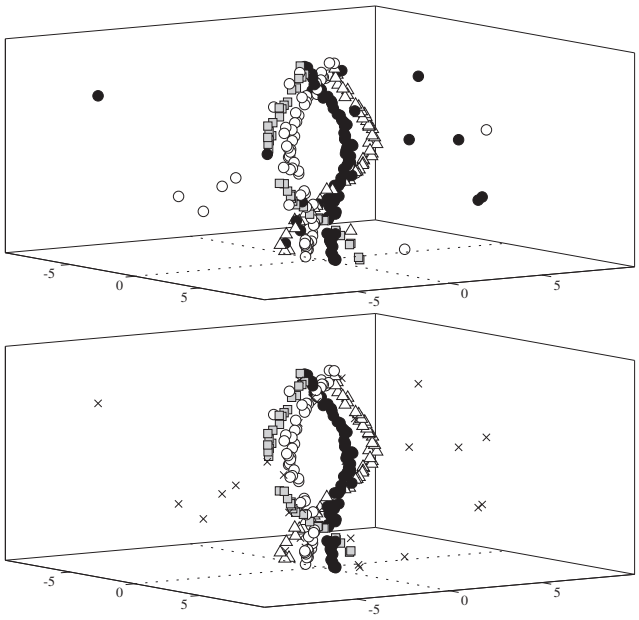


Fig. 6. Clustering with impulsive noise. Top: retrieving 4 clusters. Bottom: retrieving 5 clusters.

7. CONCLUSIONS

We presented a clustering algorithm that is capable of decoding fast flat-fading time-varying MIMO channels, marking improvements over an earlier proposed clustering-based approach.

To this end, the different steps of the spectral clustering algorithm were analyzed and adjusted to suit this particular problem better. Specifically, it was shown that the connectivity kernel is more appropriate than the Gaussian kernel in this case, and a hierarchical clustering procedure can avoid some mistakes made by k-means in the final clustering step.

Results show that the proposed method improves performance over previous clustering methods, and for high Doppler spreads it outperforms the adaptive GDFE method using less pilot symbols. Due to its use of a distance-based similarity measure, it can also be less prone to outliers.

8. REFERENCES

[1] J. Choi, H. Yu, and Y.H. Lee, "Adaptive MIMO decision feedback equalization for receivers with time-varying channels," *IEEE Trans. on Sig. Proc.*, vol. 53, no. 11, pp. 4295–4303, Nov. 2005.

[2] V. Kekatos, A.A. Rontogiannis, and K. Berberidis, "Cholesky factorization-based adaptive BLAST DFE for wideband MIMO channels," *EURASIP Journal on Adv. in Sig. Proc.*, vol. 2007, pp. 11 pages, 2007.

[3] J.R. Montalvão Filho, B. Dorizzi, and J.C. M Mota, "Channel estimation by symmetrical clustering," *IEEE Trans. on Sig. Proc.*, vol. 50, no. 6, pp. 1459–1469, June 2002.

[4] Y. Kopsinis and S. Theodoridis, "A novel cluster based MLSE equalizer for M-PAM signaling schemes," *Signal Processing*, vol. 83, no. 9, pp. 1905–1918, 2003.

[5] K. I. Diamantaras, "A clustering approach for the blind separation of multiple finite alphabet sequences from a single linear mixture," *Signal Processing*, vol. 86, no. 4, pp. 877–891, 2006.

[6] S. Van Vaerenbergh, E. Estébanez, and I. Santamaría, "A spectral clustering algorithm for decoding fast time-varying BPSK MIMO channels," in *15th European Sig. Proc. Conf. (EUSIPCO 2007)*, Poznań, Poland, Sept. 2007.

[7] S. Van Vaerenbergh and I. Santamaría, *Intelligent Systems: Techniques and Applications*, chapter A Spectral Clustering Approach for Blind Decoding of MIMO Transmissions over Time-Correlated Fading Channels, pp. 351–377, Shaker Publishing, 2008.

[8] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14 (NIPS 2002)*, Cambridge, MA, 2002, pp. 849–856, MIT Press.

[9] U. von Luxburg, "A tutorial on spectral clustering," Tech. Rep. 149, Max Planck Institute for Biological Cybernetics, August 2006.

[10] Bernd Fischer, Volker Roth, and Joachim M. Buhmann, "Clustering with the connectivity kernel," in *Advances in Neural Information Processing Systems 16 (NIPS 2004)*, Cambridge, MA, 2004, MIT Press.

[11] U. Ozertem, D. Erdogmus, and M.Á. Carreira-Perpiñán, "Density geodesics for similarity clustering," in *IEEE Int. Conf. on Acoustics, Speech and Sig. Proc. (ICASSP 2008)*, April 2008, pp. 1977–1980.

[12] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, December 1959.

[13] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems 17 (NIPS 2005)*, Cambridge, MA, 2005, pp. 1601–1608, MIT Press.

[14] T. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall PTR, 2001.